NASA TECHNICAL NOTE

NASA TN D-3762

NASA TN D-3762

A COMPUTER PROGRAM FOR CALCULATING VELOCITIES AND STREAMLINES FOR TWO-DIMENSIONAL, INCOMPRESSIBLE FLOW IN AXIAL BLADE ROWS

by Theodore Katsanis

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JANUARY 1967

NASA TN D-3762

A COMPUTER PROGRAM FOR CALCULATING VELOCITIES AND STREAMLINES FOR TWO-DIMENSIONAL, INCOMPRESSIBLE FLOW IN AXIAL BLADE ROWS

By Theodore Katsanis

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $2.50

CONTENTS

A COMPUTER PROGRAM FOR CALCULATING VELOCITIES AND STREAMLINES FOR TWO-DIMENSIONAL, INCOMPRESSIBLE FLOW IN AXIAL BLADE ROWS

by Theodore Katsanis

Lewis Research Center

SUMMARY

A FORTRAN computer program was written that gives the solution of the two-dimensional, incompressible, ideal flow problem for an infinite cascade of blades, or equivalently, a two-dimensional, circular cascade of constant radius, as in an axial-flow turbine. The computer program requires only the basic cascade geometry as input. The output includes streamline coordinates, velocity magnitude and direction throughout the passage, and the blade-surface velocities. The method is based on the stream function, with the solution of the simultaneous, linear finite-difference equations being obtained iteratively by using successive overrelaxation, with an estimated optimum overrelaxation factor.

This report includes the FORTRAN computer program that was developed, with a complete explanation of the equations involved, the method of solution, and the calculation of the velocities. A sample problem has been included to illustrate the use of the program, and to show the results that can be obtained. The program results are in good agreement with experimental results at low Mach numbers. It is concluded that reasonable results can be obtained if there is little variation in density.

INTRODUCTION

In the design of blade rows for turbines or compressors, it is desirable to obtain the velocity distribution through the passage and particularly over the blade surfaces. The trend to highly loaded blading results in more widely spaced blades with less of the passage being within a guided channel between the blades. Methods are available at present for obtaining the velocity distribution within a guided channel; however, new techniques must be developed to extend the solution to the unguided portion of the passage.

For a compressible flow problem, this solution is difficult to obtain analytically. However, a useful first approximation can be obtained based on the assumption of two-dimensional, incompressible, ideal flow for an infinite cascade of airfoils. Even though the actual velocities may not be accurate, large local variations in velocities will be evident. In addition, an approximation to the actual stagnation streamline location can be obtained, which is useful in obtaining solutions based on guided channel flow.

The two-dimensional, incompressible flow problem can be solved by finite-difference methods using the stream function, as discussed in references 1 to 4. However, the process of setting up the equations is extremely tedious, and the solution of a large number of simultaneous, linear equations requires the use of a high-speed computer. It appears at the present time that there is no generally available program which would automatically set up the equations and solve them.

For these reasons, a computer program for axial-flow blade rows has been written that requires only the basic geometry of the cascade plus upstream and downstream flow angles as input. This program obtains a numerical solution for the entire passage based on the assumption of two-dimensional, incompressible, ideal flow. Specifically, accuracy is limited to cases with small changes in density. The output includes streamline coordinates, velocity magnitude and direction throughout the passage, and blade-surface velocities. This report includes the FORTRAN computer program that was developed, with a complete explanation of the equations involved, the method of solution, and the calculation of the velocities. A stator blade of low solidity has been analyzed to illuustrate the use of the program, and these results are compared with experimental results. This report is organized so that the engineer desiring to use this program needs to read only the sections MATHEMATICAL ANALYSIS, NUMERICAL EXAMPLE AND COMPARISON WITH EXPERIMENTAL RESULTS, and DESCRIPTION OF INPUT AND OUTPUT. If a programmer is used to assist in the use of the program, all necessary information of interest to the programmer is contained in the sections DESCRIPTION OF INPUT AND OUTPUT and PROGRAM PROCEDURE.

SYMBOLS

A	coefficient matrix (eq. (A6))
a_0, a_1, a_2, a_3, a_4	coefficients in equation (A1)
B	matrix A - I
Δc	space between two closely spaced streamlines

h_1, h_2, h_3, h_4 spacing between adjacent points (eq. (A1), see also fig. 13)

I identity matrix

$\underline{k}$ constant vector (eq. (A6)), $= \begin{pmatrix} k_1 \\ \cdot \\ \cdot \\ k_n \end{pmatrix}$

L_ω coefficient matrix of equation (A7)

n number of unknown mesh points

s tangential blade spacing (fig. 2)

t thickness of stream channel

u stream function

$\underline{u}$ discrete approximation to stream function at n mesh points, $= \begin{pmatrix} u_1 \\ \cdot \\ \cdot \\ u_n \end{pmatrix}$

$\underline{u}^m$ m^{th} iterate of $\underline{u}$, $= \begin{pmatrix} u_1^m \\ \cdot \\ \cdot \\ u_n^m \end{pmatrix}$

Δu change in stream function value corresponding to Δc

V normalized fluid velocity (fig. 1)

W actual fluid velocity

w weight flow through stream channel

Δw weight flow through portion of stream channel defined by Δc

x coordinate in tangential direction (fig. 1)

z coordinate in axial direction (fig. 1)

η outer normal to region

ρ density of fluid

$\rho(\ )$ spectral radius of matrix

θ angle of flow, measured from axial

ω overrelaxation factor (eq. (A7))

Subscripts:

i	dummy variable
in	inlet or upstream
j	dummy variable
out	outlet or downstream
x	component in tangential direction (fig. 1)
z	component in axial direction (fig. 1)

Superscript:

T	transpose of vector or matrix

MATHEMATICAL ANALYSIS

It is desired to determine the flow distribution through a cascade of blades. The case considered here is a circular cascade of axial blades, such as an axial-flow turbine or compressor. It is assumed that there is no change in radius in the stream sheet, and that the radial spacing between stream sheets is constant. These assumptions make it possible to treat the flow the same as if the cascade were a two-dimensional, straight, infinite cascade (fig. 1). The flow is assumed to be steady, incompressible, nonviscous, irrotational, and isentropic. For the solution, a finite region is considered, as indicated in figure 2, with the condition that the flow along AB is the same as along HG, and the flow along CD is the same as along FE. Also, it is assumed that AH is sufficiently far upstream so that the flow is uniform along this boundary, and that the flow angle is known. Similarly, it is assumed that the flow is uniform along DE, and that the flow angle θ_{out} is known. For an actual blade row, θ_{out} may usually be determined by means of experimentally determined rules. Specifying θ_{out} along DE is mathematically equivalent to specifying the location of the stagnation point on the trailing edge of the blade.

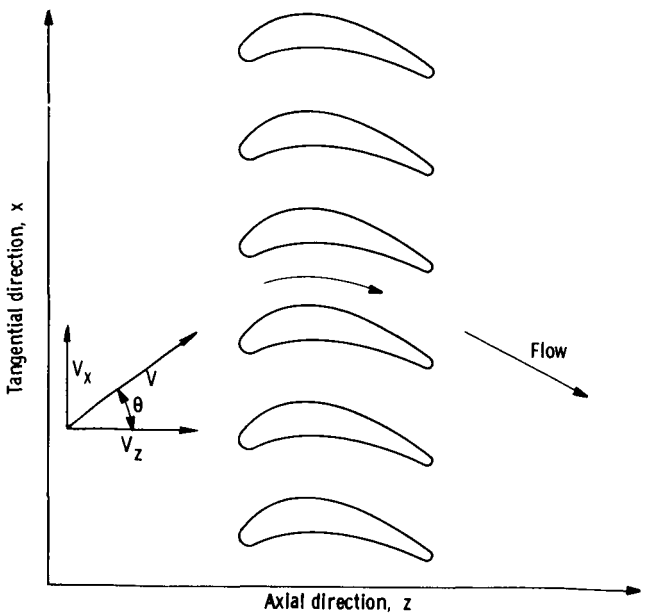


Figure 1. - Two-dimensional infinite cascade.

For the mathematical solution of the problem, the stream function is used. The assumptions that the fluid is irrotational, incompressible, and ideal, lead to the equation

$$\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial x^2} = 0 \qquad (1)$$

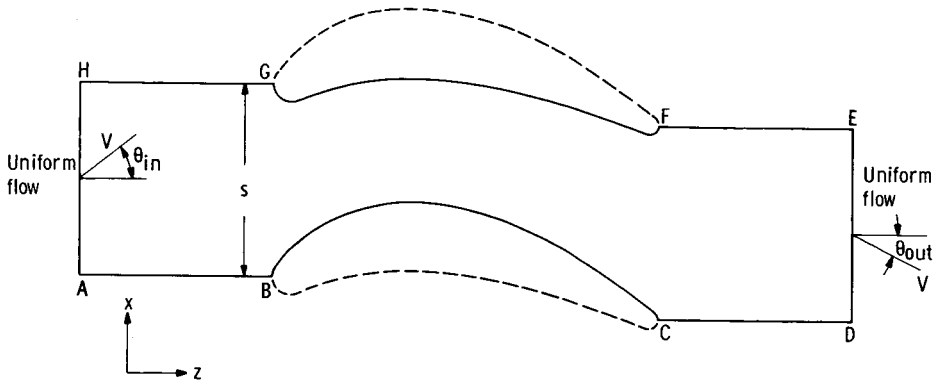


Figure 2. - Finite flow region.

Equation (1) has a unique solution satisfying the boundary conditions implied by the physical conditions mentioned in the preceding paragraph.

The stream function is defined to be equal to zero along BC and equal to 1 along GF. For a normalized flow of 1 unit of volume per unit of time, with a unit thickness of stream channel, the normalized velocity components are given by

$$\left.\begin{aligned} V_x &= -\frac{\partial u}{\partial z} \\ V_z &= \frac{\partial u}{\partial x} \end{aligned}\right\} \qquad (2)$$

All velocities obtained are normalized velocities.

Since equation (1) is elliptic, boundary conditions for the stream function on the entire boundary ABCDEFGH are required. Along BC, $u = 0$; along FG, $u = 1$. Along AB, GH, CD, and EF, a periodic condition exists; that is, the value of u along HG and FE is exactly 1 greater than it is along AB and CD. Along AH and DE, $\partial u/\partial \eta$ is known, where η is in the direction of the outer normal. It can be seen that $V_z = \partial u/\partial x = [u(H) - u(A)]/s = 1/s$ along AH and DE. Since $V_x/V_z = \tan \theta$, where θ is the flow angle, it can be used in equation (2) with the following result:

$$\left.\begin{aligned} \left(\frac{\partial u}{\partial \eta}\right)_{in} &= \frac{\tan \theta_{in}}{s} \quad \text{along AH} \\ \left(\frac{\partial u}{\partial \eta}\right)_{out} &= -\frac{\tan \theta_{out}}{s} \quad \text{along DE} \end{aligned}\right\} \qquad (3)$$

These are the required boundary conditions to determine a solution to equation (1). After computing a numerical solution to equation (1) in a given flow region, the velocity at any point can be computed from equation (2) by using numerical differentiation. The streamlines are located by the contours of equal stream-function values. The method used for the numerical solution of equation (1) is described in appendixes A and B.

To obtain an actual velocity W, three more quantities must be known, the density ρ, the thickness of the stream channel t, and the corresponding weight flow w, flowing in the stream channel. The normalized velocity V, between two closely spaced streamlines, with space Δc, is given by $V = \Delta u/\Delta c$. The actual velocity is given by $W = \Delta w/(\rho t\, \Delta c)$. Since $\Delta u = \Delta w/w$, the actual velocity can be written as

$$W = \frac{w}{\rho t} V \tag{4}$$

Thus, the actual velocity can be obtained by multiplying V by the factor $w/\rho t$.

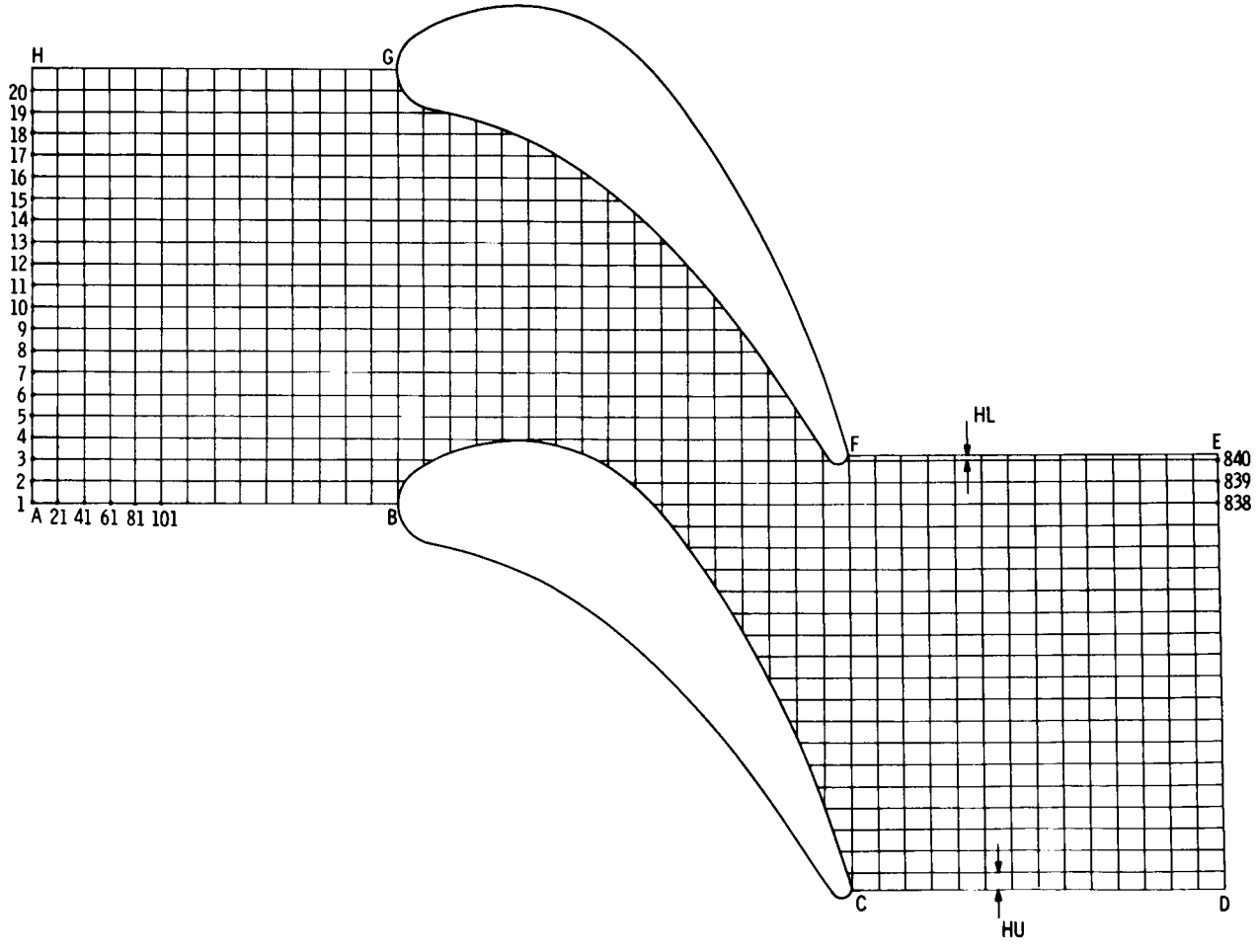


Figure 3. - Mesh used for numerical example. Numbers are mesh point indexes (I in program). There are 840 unknown mesh points.

TABLE I. - INPUT FORM WITH DATA FOR NUMERICAL EXAMPLE

[Top row of numbers are card column numbers.]

1 10	11 20	21 30	31 40	41 50	51 60	61 70	71 80
PITCH	CHORD	STGR	THETAI	THETAO	DTLR		
1.6336	1.679	-1.451	0.	-67.	.0001		

1 10	11 20	21 30	31 40	41 50	51 60
RI	ALUI	ALLI	RO	ALUO	ALLO
.15	28.3	-14.2	.035	-72.4	-56.1

1 5	6 10	11 15	16 20	21 25	26 30	31 35
MXBI	MXBO	MX	NBBI	NUSP	NLSP	NINT
15	32	46	20	7	6	5

1 10	11 20	21 30	31 40	41 50	51 60	61 70
ZU ARRAY →						
	.3376	.6752	1.0128	1.3504	1.5192	

1 10	11 20	21 30	31 40	41 50	51 60	61 70
XSPU ARRAY →						
	.230	.200	-.069	-.605	-.962	

1 10	11 20	21 30	31 40	41 50	51 60
ZL ARRAY →					
	.3376	.6752	1.0128	1.3504	

1 10	11 20	21 30	31 40	41 50	51 60
XSPL ARRAY →					
	1.4305	1.2626	.9745	.5611	

1 5	6 10	11 15	16 20	21 25	26 30	31 35	36 40	41 45
BLDATA	NULAKI	ERPRT	STRFN	SLCRD	SLPLT	ARPRT	INTVEL	SURVEL
1	1	1	1	1	1	1	1	1

1 10	11 20	21 30	31 40	41 50
W	WR	TOLER	BDA	BDD
0.	.00001	.000001	0.	2.

TABLE II. - STREAMLINE COORDINATES

z-coordinate	Stream function	x-coordinate	Stream function	x-coordinate	Stream function	x-coordinate
-1.3827059	0.2000000	0.2828171	0.4000000	0.6087984	0.6000000	0.9343205
	0.8000000	1.2610173	1.0000000	1.5889378		
-1.2839411	0.2000000	0.2828164	0.4000000	0.6087982	0.6000000	0.9343207
	0.8000000	1.2610173	1.0000000	1.5889375		
-1.1851764	0.2000000	0.2829707	0.4000000	0.6088455	0.6000000	0.9341995
	0.8000000	1.2608900	1.0000000	1.5889825		
-1.0864117	0.2000000	0.2833027	0.4000000	0.6089464	0.6000000	0.9339404
	0.8000000	1.2606169	1.0000000	1.5890793		
-0.9876470	0.2000000	0.2838612	0.4000000	0.6091133	0.6000000	0.9335084
	0.8000000	1.2601577	1.0000000	1.5892419		
-0.8888823	0.2000000	0.2847287	0.4000000	0.6093656	0.6000000	0.9328464
	0.8000000	1.2594437	1.0000000	1.5894939		
-0.7901176	0.2000000	0.2860345	0.4000000	0.6097304	0.6000000	0.9318694
	0.8000000	1.2583671	1.0000000	1.5898719		
-0.6913529	0.2000000	0.2879749	0.4000000	0.6102414	0.6000000	0.9304561
	0.8000000	1.2567629	1.0000000	1.5904310		
-0.5925882	0.2000000	0.2908439	0.4000000	0.6109357	0.6000000	0.9284417
	0.8000000	1.2543811	1.0000000	1.5912533		
-0.4938235	0.2000000	0.2950752	0.4000000	0.6118429	0.6000000	0.9256102
	0.8000000	1.2508475	1.0000000	1.5924602		
-0.3950588	0.2000000	0.3012871	0.4000000	0.6129625	0.6000000	0.9216915
	0.8000000	1.2456138	1.0000000	1.5942300		
-0.2962940	0.2000000	0.3102825	0.4000000	0.6142236	0.6000000	0.9163676
	0.8000000	1.2379232	1.0000000	1.5968190		
-0.1975293	0.2000000	0.3228580	0.4000000	0.6154226	0.6000000	0.9092884
	0.8000000	1.2268676	1.0000000	1.6005786		
-0.9876463E-01	0.2000000	0.3391626	0.4000000	0.6161498	0.6000000	0.9000907
	0.8000000	1.2116756	1.0000000	1.6061686		
0.7264316E-07	0	0	0.2000000	0.3572025	0.4000000	0.6157414
	0.6000000	0.8883994	0.8000000	1.1919836	1.0000000	1.6335998
0.9876478E-01	0	0.1425575	0.2000000	0.3754763	0.4000000	0.6132330
	0.6000000	0.8737967	0.8000000	1.1691997	1.0000000	1.4926214
0.1975295	0	0.1878169	0.2000000	0.3887673	0.4000000	0.6075340
	0.6000000	0.8557513	0.8000000	1.1430696	1.0000000	1.4678576
0.2962942	0	0.2204743	0.2000000	0.3956983	0.4000000	0.5974513
	0.6000000	0.8335885	0.8000000	1.1130518	1.0000000	1.4428377
0.3950589	0	0.2388816	0.2000000	0.3946070	0.4000000	0.5817730
	0.6000000	0.8065076	0.8000000	1.0780016	1.0000000	1.4104352
0.4938236	0	0.2414688	0.2000000	0.3838285	0.4000000	0.5592464
	0.6000000	0.7736225	0.8000000	1.0368844	1.0000000	1.3677690
0.5925883	0	0.2267293	0.2000000	0.3616488	0.4000000	0.5285581
	0.6000000	0.7339860	0.8000000	0.9888542	1.0000000	1.3147966
0.6913530	0	0.1931579	0.2000000	0.3264130	0.4000000	0.4883344
	0.6000000	0.6865851	0.8000000	0.9331863	1.0000000	1.2515562
0.7901177	0	0.1390832	0.2000000	0.2765197	0.4000000	0.4371329
	0.6000000	0.6303272	0.8000000	0.8691821	1.0000000	1.1780625
0.8888824	0	0.6242253E-01	0.2000000	0.2101129	0.4000000	0.3734106
	0.6000000	0.5640146	0.8000000	0.7960975	1.0000000	1.0942724
0.9876471	0	-0.3896921E-01	0.2000000	0.1251964	0.4000000	0.2955329
	0.6000000	0.4863461	0.8000000	0.7130878	1.0000000	1.0001342
1.0864118	0	-0.1666834	0.2000000	0.1996823E-01	0.4000000	0.2019297
	0.6000000	0.3960013	0.8000000	0.6191841	1.0000000	0.8955583
1.1851765	0	-0.3174163	0.2000000	-0.1065226	0.4000000	0.9149652E-01
	0.6000000	0.2919446	0.8000000	0.5134198	1.0000000	0.7801226
1.2839412	0	-0.4853115	0.2000000	-0.2536652	0.4000000	-0.3619862E-01
	0.6000000	0.1740957	0.8000000	0.3952740	1.0000000	0.6532318
1.3827059	0	-0.6647861	0.2000000	-0.4196129	0.4000000	-0.1821745
	0.6000000	0.4376124E-01	0.8000000	0.2659561	1.0000000	0.5143370
1.4814706	0	-0.8688387	0.2000000	-0.6027053	0.4000000	-0.3480026
	0.6000000	-0.1015228	0.8000000	0.1305427	1.0000000	0.3659094
1.5802353	0	-1.1354732	0.2000000	-0.8090605	0.4000000	-0.5334354
	0.6000000	-0.2686955	0.8000000	-0.8672454E-02	1.0000000	0.2151804
1.6790000	0	-1.4509354	0.2000000	-1.0549767	0.4000000	-0.7401864
	0.6000000	-0.4571657	0.8000000	-0.1790342	1.0000000	0.1816985
1.7777647	0.2000000	-1.3349311	0.4000000	-0.9744606	0.6000000	-0.6645459
	0.8000000	-0.3740317	1.0000000	-0.7852054E-01		
1.8765294	0.4000000	-1.2328247	0.6000000	-0.8922524	0.8000000	-0.5834380
	1.0000000	-0.2844415	1.2000000	0.3226341E-01		
1.9752941	0.6000000	-1.1376492	0.8000000	-0.8078439	1.0000000	-0.4977273
	1.2000000	-0.1894631	1.4000000	0.1428416		
2.0740588	0.6000000	-1.3871769	0.8000000	-1.0454709	1.0000000	-0.7211562
	1.2000000	-0.4082030	1.4000000	-0.9117272E-01		
2.1728235	0.8000000	-1.2885423	1.0000000	-0.9543028	1.2000000	-0.6323167
	1.4000000	-0.3157693	1.5999999	0.7962946E-02		
2.2715882	1.0000000	-1.1927978	1.2000000	-0.8631470	1.4000000	-0.5415961
	1.6000000	-0.2214419	1.8000000	0.1062508		
2.3703529	1.0000000	-1.4304440	1.2000000	-1.0985718	1.4000000	-0.7715363
	1.6000000	-0.4493716	1.8000000	-0.1261838		
2.4691176	1.2000000	-1.3347470	1.4000000	-1.0050443	1.6000000	-0.6793078
	1.8000000	-0.3560725	1.9999999	-0.3071414E-01		
2.5678823	1.4000000	-1.2398961	1.6000000	-0.9117384	1.8000000	-0.5864756
	1.9999999	-0.2621114	2.1999999	0.6456540E-01		
2.6666470	1.6000000	-1.1455710	1.8000000	-0.8184018	2.0000000	-0.4931497
	2.1999999	-0.1678234	2.3999999	0.1595169		
2.7654116	1.6000000	-1.3794466	1.8000000	-1.0515175	2.0000000	-0.7249289
	2.1999999	-0.3994780	2.3999999	-0.7343548E-01		
2.8641763	1.8000000	-1.2850309	2.0000000	-0.9575694	2.2000000	-0.6313061
	2.3999999	-0.3056053	2.5999999	0.2093099E-01		
2.9629410	2.0000000	-1.1907387	2.2000000	-0.8636367	2.3999999	-0.5375749
	2.5999999	-0.2116467	2.7999999	0.1152406		
3.0617057	2.0000000	-1.4240814	2.2000000	-1.0964792	2.3999999	-0.7696929
	2.5999999	-0.4438053	2.7999999	-0.1176743		

TABLE III. - SURFACE VELOCITIES

(a) Based on axial components

Upper surface				Lower surface		
z-coordinate	Velocity, W	Angle, θ, deg	Axial component of velocity, W_z	Velocity, W	Angle, θ, deg	Axial component of velocity, W_z
0.7264E-07	0	90.00	0.4779E-02	0	-90.00	-0.2898E-01
0.9876E-01	0.9620	27.32	0.8547	0.7337	-20.67	0.6896
0.1975	1.0874	21.67	1.0106	0.6268	-13.81	0.6099
0.2963	1.2478	14.64	1.2073	0.5901	-16.22	0.5682
0.3951	1.4228	6.21	1.4144	0.5779	-21.52	0.5403
0.4938	1.5867	-3.38	1.5840	0.5939	-26.74	0.5345
0.5926	1.7093	-13.61	1.6613	0.6184	-31.54	0.5330
0.6914	1.7863	-23.77	1.6348	0.6616	-35.90	0.5440
0.7901	1.8420	-33.35	1.5386	0.7157	-39.87	0.5599
0.8889	1.8592	-41.92	1.3835	0.7816	-43.48	0.5807
0.9876	1.8415	-49.21	1.2030	0.8590	-46.74	0.6056
1.0864	1.7827	-54.84	1.0267	0.9566	-49.73	0.6394
1.1852	1.7377	-58.36	0.9115	1.0566	-52.57	0.6679
1.2839	1.7094	-60.49	0.8419	1.1767	-55.23	0.7021
1.3827	1.7409	-61.98	0.8179	1.2999	-57.63	0.7327
1.4815	1.7970	-66.76	0.7090	1.4380	-58.77	0.7877
1.5802	1.6887	-71.69	0.5304	1.5345	-58.49	0.8465
1.6790	0	-90.00	0.4220	0	90.00	-0.3237

(b) Based on tangential components

Upper surface				Lower surface			
z-coordinate	Velocity, W	Angle, θ, deg	Tangential component of velocity, W_x	z-coordinate	Velocity, W	Angle, θ, deg	Tangential component of velocity, W_x
0.7264E-07	0.1141	90.00	0.1141	0.2424E-01	0.4755	-56.97	-0.3987
0.2424E-01	0.7534	56.97	0.6316	0.1875	0.4767	-13.31	-0.1097
0.1412	0.9611	25.05	0.4069	0.4485	0.5761	-23.57	-0.2304
0.7505	1.8539	-29.61	-0.9159	0.6058	0.6241	-31.06	-0.3220
0.8667	1.8544	-40.10	-1.1944	0.7283	0.6747	-36.17	-0.3982
0.9526	1.8461	-46.77	-1.3451	0.8323	0.7350	-40.05	-0.4730
1.0230	1.8122	-51.50	-1.4183	0.9241	0.8016	-43.17	-0.5484
1.0841	1.7829	-54.73	-1.4557	1.0073	0.8727	-45.75	-0.6252
1.1393	1.7448	-56.93	-1.4622	1.0838	0.9356	-47.98	-0.6951
1.1909	1.7195	-58.52	-1.4663	1.1548	1.0111	-49.97	-0.7742
1.2397	1.7101	-59.68	-1.4762	1.2212	1.0886	-51.75	-0.8549
1.2866	1.7040	-60.54	-1.4836	1.2838	1.1574	-53.36	-0.9287
1.3322	1.7113	-61.16	-1.4990	1.3429	1.2352	-54.82	-1.0096
1.3766	1.7362	-61.81	-1.5303	1.3992	1.3175	-55.97	-1.0919
1.4191	1.7699	-63.34	-1.5818	1.4536	1.3867	-56.62	-1.1580
1.4584	1.7835	-65.39	-1.6215	1.5071	1.4612	-56.85	-1.2233
1.4939	1.7700	-67.53	-1.6356	1.5606	1.5188	-56.68	-1.2692
1.5260	1.7468	-69.52	-1.6364	1.6148	1.6407	-56.10	-1.3618
1.5553	1.7115	-70.90	-1.6174	1.6732	0.7964	56.55	-0.6645
1.5829	1.6632	-71.76	-1.5796				
1.6094	1.6281	-72.27	-1.5508				
1.6353	1.5962	-72.51	-1.5224				
1.6610	1.5904	-72.51	-1.5169				
1.6790	1.4800	-90.00	-1.4800				

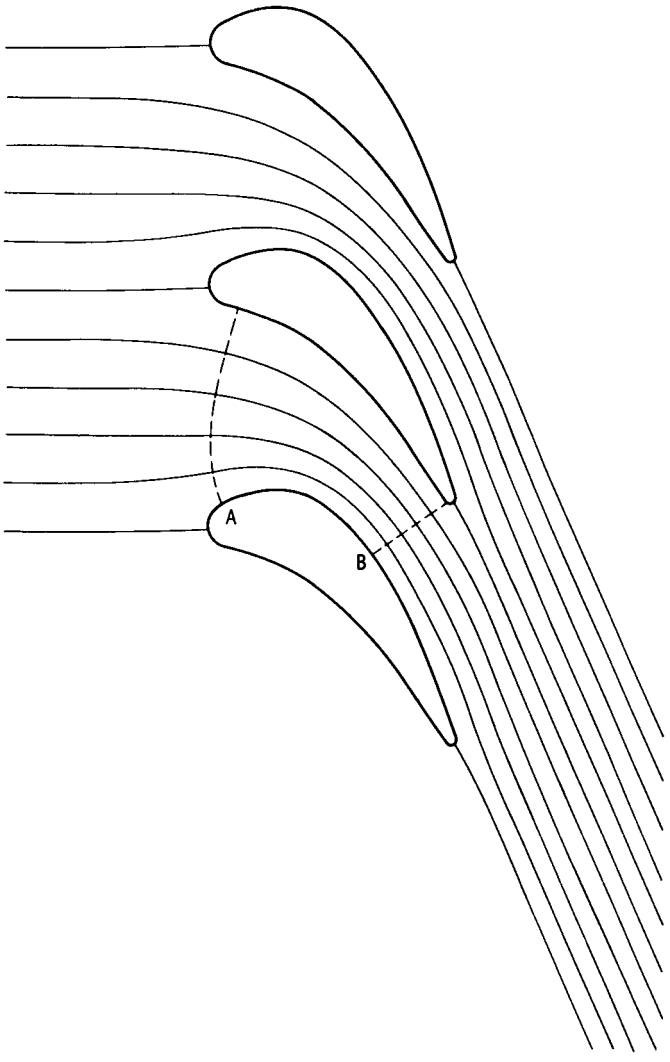


Figure 4. - Streamlines for numerical example.

• NUMERICAL EXAMPLE AND COMPARISON WITH EXPERIMENTAL RESULTS

A numerical example, based on a stator nozzle mean blade section (ref. 5), is given to illustrate the results and conclusions that may be obtained by the use of the program. The blade shape is shown in figure 3, and the actual input in table I. The execution time for this example was less than 1 minute.

The streamlines obtained by the program, shown in figure 4, were plotted from the data in table II, which is output from the program. If the incompressible stagnation streamlines are assumed to be close to the stagnation streamlines for compressible flow, this information is useful in obtaining a compressible flow analysis by other methods. The beginning and the end of the guided channel are indicated by A and B in figure 4.

The surface velocities obtained from the program are given in table III and are plotted against axial distance in figure 5. The curve for each blade surface (upper or lower) is plotted from two sets of velocity data, one based on axial velocities alone and blade surface angle, and the other based on tangential velocities alone and blade surface angle. A velocity based on a component nearly normal to a surface cannot be expected to be accurate, so that the curve shown favors the velocity based on the component most nearly parallel to the blade surface. For intermediate blade surface angles (e.g., between 30° and 60° from axial) the difference between the two velocities is small. The velocities must be

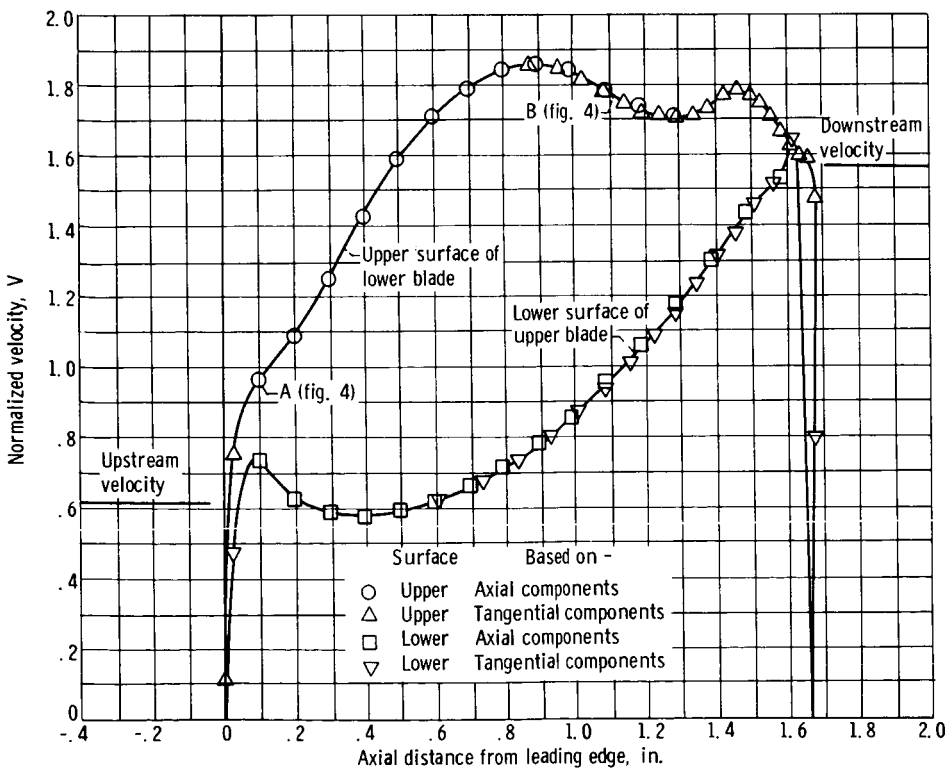
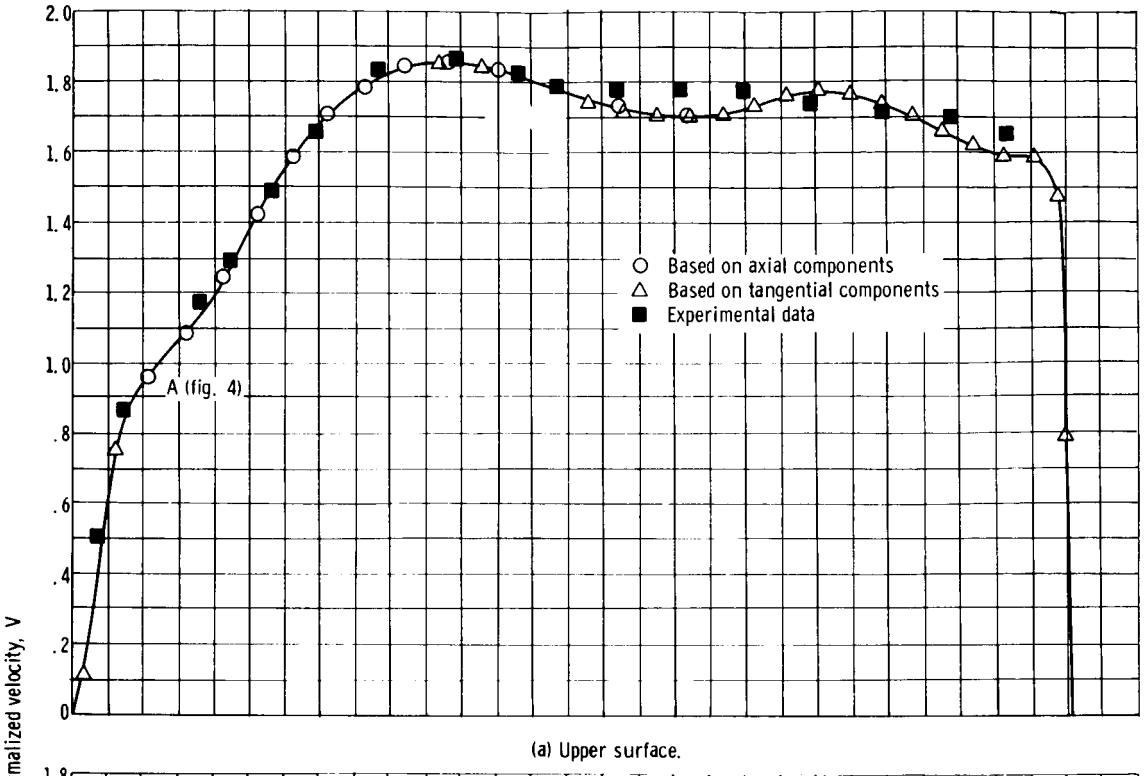
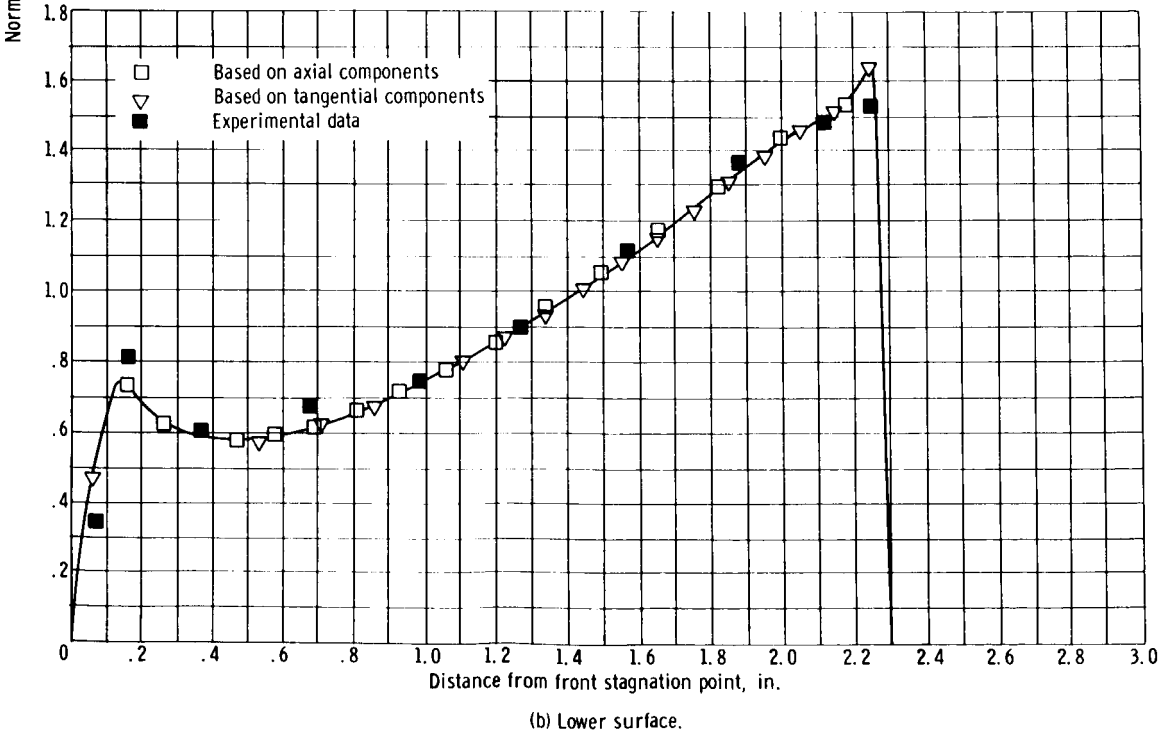


Figure 5. - Blade surface velocities for numerical example.



(a) Upper surface.



(b) Lower surface.

Figure 6. - Blade surface velocity for numerical example.

zero at the stagnation points. The zero blade-surface velocity is located at the stagnation point, which is an extrapolation from calculated points on the stagnation streamline in figure 4.

In figure 6 the surface velocity data are plotted against blade-surface length from the front stagnation point. This type of relation gives a most accurate picture of the acceleration along the surface. Experimental data for this blade section at an exit Mach number of 0.33 are also plotted in figure 6. Equation (4) was used to correct the experimental data to the normalized velocities shown in figure 6. There is generally good agreement with the computed values for this example.

Contours of equal-velocity magnitudes are shown in figure 7 plotted from the output data giving the velocities at each mesh point. The diagram of velocity contours shows the velocity distribution through the passageway. In this case, the flow is accelerating fairly evenly through the passage, as is desired in a stator nozzle. Also, there is a substantial variation in velocity across the passage width at the trailing edge of the blade.

In this example, the mesh region was extended 14 mesh spaces axially, both upstream and downstream of the blade. The effect of varying this distance on the blade surface velocities was checked. There was no detectable change when the extension was reduced to 9 mesh spaces and only a slight change when it was reduced to 4 mesh spaces.

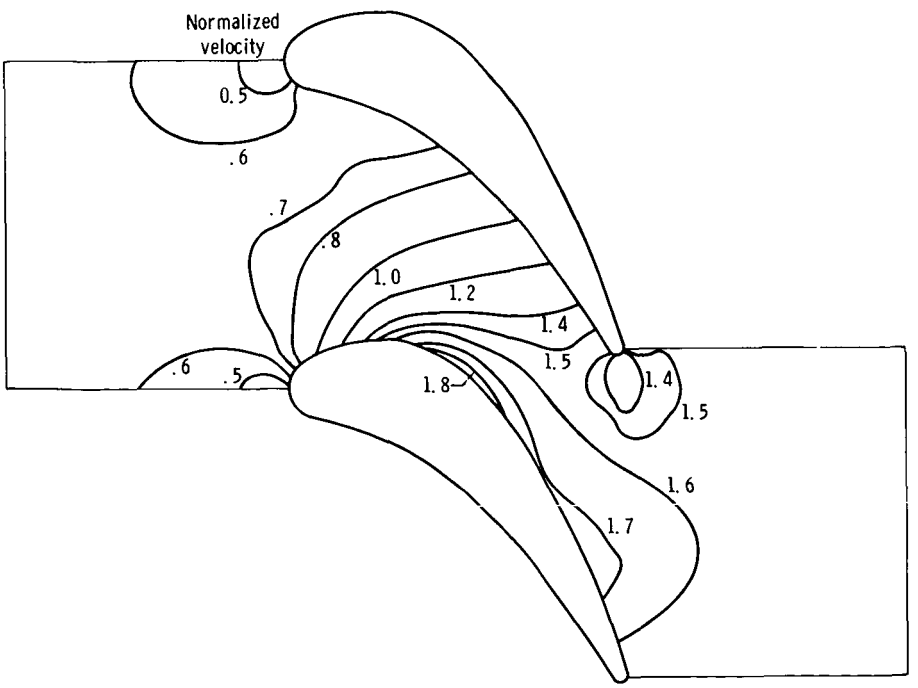


Figure 7. - Normalized velocity contours for numerical example.

DESCRIPTION OF INPUT AND OUTPUT

The computer program requires as input sufficient information to describe the blade shape accurately, the inlet and outlet angles, the extent of the region to be considered, and the mesh size to be used. Also required are values for convergence criteria, an estimate for an overrelaxation factor, if available, and data indicating what output is desired. Output obtained from the program includes streamline locations, velocity magnitude and direction at all interior points, and the blade-surface velocities.

Input

Table I (p. 7) shows the input variables required as they are to be punched on the data cards. There are two types of input variables, geometric and nongeometric. The geometric input variables are shown in figure 8.

The blade shape is defined by specifying the leading- and trailing-edge radii and a sufficient number of blade-surface coordinates so that a cubic spline curve through these points will specify the blade shape adequately. In addition, the slopes at the end points are specified to be tangent to the leading- and trailing-edge radii. A cubic spline curve, which is a piecewise cubic polynomial, is a mathematical expression for the shape taken

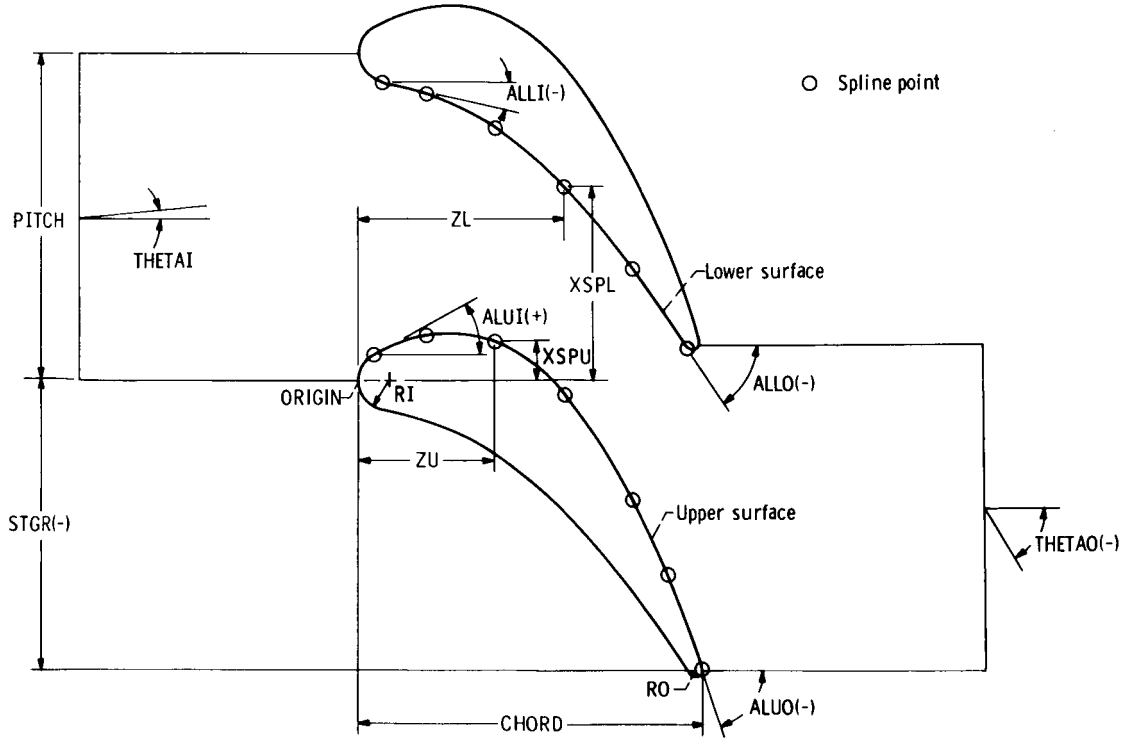


Figure 8. - Geometric variables required as input.

by an idealized spline passing through the given points. Reference 6 describes spline curves and a method for determining the equation of the spline curve. Using this method requires few points to specify most blade shapes accurately, usually no more than five or six points (and often less), in addition to the two end points. As a guide, enough points should be specified so that a physical spline passing through these points would accurately follow the blade shape. This means that the spline points should be closer where there is large curvature and farther apart where there is small curvature. The coordinates of the spline points are given with respect to the leading edge of the lower blade, as shown in figure 8. The standard sign convention is used for angles, as indicated in figure 8, and the blade should be oriented with the concave side down.

The card format for the nongeometric variables is shown in table I (p. 7). All the variables on the card starting with BLDATA are used to indicate what output is desired. A value of zero for any of these variables will cause the output associated with that variable to be omitted.

The mesh spacing and upstream and downstream regions are determined by the values of MXBI, MXBO, MX, and NBBI. The mesh spacing must be chosen so that there are not more than 2500 unknown mesh points. If there are more than 2500 mesh points, there will be an error return and a print out of the number of mesh points. The nongeometric input variables are as follows:

DTLR Tolerance for mesh points near boundary mesh points. If a mesh point is closer than DTLR to the boundary, the boundary is considered to go through the mesh point. A suggested value is approximately 0.001 times the basic mesh spacing.

MXBI Number of mesh spaces between AH and BG (fig. 3).

MXBO Number of mesh spaces between AH and CF (fig. 3).

MX Total number of mesh spaces in z-direction between AH and DE; maximum of 100 (fig. 3).

NBBI Number of mesh spaces between AB and HG, maximum of 50 (fig. 3).

The variables MXBI, MXBO, and MX specify the mesh region in the z-direction. The difference MXBO - MXBI is the number of mesh spaces in the axial chord length. Then the values of MXBI and the difference MX - MXBO determine the distance upstream and downstream included in the region of the solution. NBBI is the number of mesh spaces desired in the x-direction between A and H. All the numbers on this card and on the card beginning with BLDATA are integers (no decimal point) and must be right adjusted.

NUSP Number of spline points including end points that are tangent to leading- and trailing-edge radii for the upper surface (BC) of the blade (figs. 3 and 8).

NLSP — Same as NUSP, except for the lower surface (GF) of the blade.

NINT — Number of streamlines desired as output; maximum of 10.

BLDATA — A value of 1 will result in first and second derivatives at the spline points, and also the x-coordinate at each mesh line for each blade surface being printed as output; zero will cause this output to be omitted.

NULAKI — A value of 1 will cause the values of NU and NL, which are internal variable names, the coefficient array A, the vector K, and also the value of I for the adjacent points, I1, I2, I3, and I4 to be printed out; zero will cause this output to be omitted. This information is needed only for debugging.

ERPRT — A value of 1 will result in printing out the maximum change in the stream function for each iteration of the SOR equation (eq. (A7)); zero will cause this output to be omitted.

STRFN — A value of 1 will result in printing out the value of the stream function at each mesh point in the region; zero will cause this output to be omitted.

SLCRD — A value of 1 will result in printing out the streamline coordinates at each vertical mesh line; zero will cause this output to be omitted.

SLPLT — A value of 1 will result in printing out a plot of the streamlines; zero will cause this output to be omitted.

ARPRT — A value of 1 will result in printing out the values of the normalized axial velocity components at each mesh point and at each vertical mesh line along the blade surfaces, the values of the normalized tangential velocity component at each mesh point, and the z-coordinate and tangential velocity component at each horizontal mesh line along the blade surfaces; zero will cause this output to be omitted.

INTVEL — A value of 1 will result in printing out the velocity magnitude and the flow angle at each mesh point; zero will cause this output to be omitted.

SURVEL — A value of 1 will result in printing out the z-coordinate, surface velocity, and flow angle for each blade surface, based separately on axial velocity components and on tangential velocity components; zero will cause this output to be omitted.

W — A value for overrelaxation factor ω to be used in equation (A7); if W = 0, the program will calculate an estimated value for optimum overrelaxation factor (see appendixes A and B for discussion).

WR — A tolerance specified for the calculation of overrelaxation factor ω; a suggested value is 10^{-5}.

TOLER When the maximum absolute value of the change in the stream function is less than TOLER, the SOR iteration is considered converged; a suggested value is 10^{-6}.

BDA Estimated value of the stream function at A (fig. 3). The value zero may be used.

BDD Estimated value of the stream function at D. The value zero may be used.

Output

An example of the output for the sample problem is given in table IV. This problem is the same as for the numerical example, but with a greatly reduced number of mesh points to reduce the amount of output. The only items that have been changed are MXBI = 5, MXBO = 10, MX = 15, and NBBI = 5. Each section of the output has been numbered to correspond to the following descriptions.

(1) The first output of the program is a listing of the input data. The output obtained after this depends on what is specified to be printed out.

(2) This output is the number of unknown mesh points.

(3) If SLCRD = 1, the streamline coordinates along each vertical mesh line are printed. The number of streamlines is given by the value of NINT as input, and there may be as many as 10. If SLCRD = 0, this output is omitted.

(4) If INTVEL = 1, the velocity magnitudes and flow angles will be printed at each unknown mesh point, grouped by vertical mesh lines. For each vertical line, the values are given starting at the lower boundary and then for increasing values of x. If INTVEL = 0, this output will be omitted.

(5) If SURVEL = 1, the surface velocities based on axial velocity components are printed, followed by the surface velocities based on tangential components. If SURVEL = 0, this output is omitted.

The foregoing items give the basic output desired. In addition, further detail and information useful for debugging are given by the following items.

(6) If BLDATA = 1 as input, a listing of the spline points for the upper and lower blade surfaces with the first and second derivatives is printed. This is followed by the x-coordinates at each vertical mesh line for the upper blade surface and then for the lower blade surface. If BLDATA = 0, this output is omitted.

(7) If NULAKI = 1 as input, the NU and NL arrays are printed, followed by the A and K arrays, with each corresponding value for IA, I, I1, I2, I3, and I4. If NULAKI = 0, this output is omitted.

(8) If ERPRT = 1 as input, the maximum change of any value of the stream-function

estimate for each iteration is printed out as a measure of the error. If ERPRT = 0, this output is omitted.

(9) If STRFN = 1, the stream-function value at each unknown mesh point is printed. If STRFN = 0, this output is omitted.

(10) If SLPLT = 1, there will be a printer plot of the streamline coordinates. Because of the limitations of the printer, the plot is not accurate, but will give an idea of how the streamlines look and is a check on whether any errors have been made in the input data. If SLPLT = 0, this output is omitted.

(11) If ARPRT = 1, the axial velocity components (WZ ARRAY) for each unknown mesh point will be printed, grouped by vertical mesh lines. This is followed by axial velocity components along the upper surface (WZU ARRAY) at each vertical mesh line, and then the axial velocity components along the lower surface (WZL ARRAY) at each vertical mesh line. This is followed by the tangential velocity components (WX ARRAY) for each unknown mesh point, grouped by vertical mesh lines. Next is the tangential velocity components along the upper surface (WXU) at each horizontal mesh line, together with the corresponding z-coordinate (ZXU) and then the same information along the lower surface (ZXL and WXL). If ARPRT = 0, this output will be omitted.

(12) If an estimate for the overrelaxation parameter ω is not supplied, an estimate will be calculated by the program, and for each iteration the estimated upper and lower bounds for the optimum overrelaxation factor and for $\rho(L_1)$ will be printed out. These bounds cannot be expected to converge to the same value in general (within practical time limits) for reasons explained in appendix B.

TABLE IV. - SAMPLE OUTPUT

Description (a)

1

```
  1.6336000        1.6790000       -1.4510000        0               -67.000000        0.1000000E-03
  0.1500000        28.300000       -14.200000        0.3500000E-01   -72.400000       -56.100000
   5   10   15     5    7    6     5
 -0                 0.3376000        0.6752000        1.0128000        1.3504000        1.5192000       -0
 -0                 0.2300000        0.2000000       -0.6900000E-01   -0.6050000       -0.9620000      -0
 -0                 0.3376000        0.6752000        1.0128000        1.3504000       -0
 -0                 1.4305000        1.2626000        0.9745000        0.5611000       -0
  1    1    1     1    1    1     1    1    1
  0                 0.1000000E-04    0.1000000E-05    0                2.0000000
```

6

```
NO. OF POINTS =  7
        Z                    X                DERIVATIVE          2ND DERIV.
    0.78886758E-01       0.13207159           0.53844462         -1.08831435
    0.33760000           0.23000000           0.19945451         -1.53227139
    0.67520000           0.20000000          -0.40684929         -2.05957577
    1.01279999          -0.69000000E-01      -1.22904786         -2.81126899
    1.35040000          -0.60500000          -1.83039498         -0.75121377
    1.51920000          -0.96200000          -2.62059435         -8.61133850
    1.67736167          -1.44041702          -3.15240154          1.88648333
NO. OF POINTS =  6
        Z                    X                DERIVATIVE          2ND DERIV.
    0.11320388           1.48818319          -0.25303894          0.47763922
    0.33760000           1.43050000          -0.31869122         -1.06278561
    0.67520000           1.26260000          -0.67522165         -1.04936150
    1.01279999           0.97450000          -1.03255486         -1.06754149
    1.35040000           0.56110000          -1.42826733         -1.27672654
    1.61494957           0.16307892          -1.48815751          0.82395561
INTERPOLATED COORDINATES AND SLOPES COMPUTED BY BLDCRD
     0.                 -0.84198380E 00
     0.22963849E 00      0.20220982E 00
     0.20145132E 00     -0.39944493E 00
    -0.62404067E-01     -0.12138995E 01
    -0.59184095E 00     -0.18248280E 01
    -0.14509543E 01     -0.16329249E 01
INTERPOLATED COORDINATES AND SLOPES COMPUTED BY BLDCRD
     0.16336000E 01     -0.73103970E 00
     0.14310719E 01     -0.31678933E 00
     0.12650240E 01     -0.67144369E 00
     0.98006022E 00     -0.10267909E 01
     0.57135049E 00     -0.14190909E 01
     0.18255432E 00     -0.11903937E 01
```

2

```
NUMBER OF INTERIOR MESH POINTS =   72
```

[a]See pp. 17 and 18.

TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)					
	LIST OF NU AND NL				
	0	4			
	0	4			
	0	4			
	0	4			
	1	4			
	1	4			
	1	3			
4	-0	2			
	-1	1			
	-4	0			
	-5	0			
	-5	0			
	-5	0			
	-5	0			
	-5	0			
	WMAX = 2.000000	WMIN = 1.040602	LMAX = 1.000000	LMIN = 0.149983	
	WMAX = 1.999827	WMIN = 1.116348	LMAX = 1.000000	LMIN = 0.373439	
	WMAX = 1.999701	WMIN = 1.123316	LMAX = 1.000000	LMIN = 0.390908	
	WMAX = 1.905932	WMIN = 1.177113	LMAX = 0.997564	LMIN = 0.511298	
	WMAX = 1.803122	WMIN = 1.246356	LMAX = 0.988078	LMIN = 0.634364	
	WMAX = 1.753757	WMIN = 1.305510	LMAX = 0.980285	LMIN = 0.717010	
	WMAX = 1.737053	WMIN = 1.364390	LMAX = 0.977085	LMIN = 0.782978	
	WMAX = 1.724705	WMIN = 1.415331	LMAX = 0.974522	LMIN = 0.829351	
	WMAX = 1.715560	WMIN = 1.460738	LMAX = 0.972510	LMIN = 0.863713	
	WMAX = 1.708773	WMIN = 1.498849	LMAX = 0.970953	LMIN = 0.888205	
	WMAX = 1.703722	WMIN = 1.529223	LMAX = 0.969759	LMIN = 0.905226	
	WMAX = 1.699950	WMIN = 1.552430	LMAX = 0.968846	LMIN = 0.916882	
	WMAX = 1.697124	WMIN = 1.569585	LMAX = 0.968150	LMIN = 0.924802	
	WMAX = 1.694999	WMIN = 1.581955	LMAX = 0.967621	LMIN = 0.930167	
	WMAX = 1.693396	WMIN = 1.590714	LMAX = 0.967218	LMIN = 0.933798	
	WMAX = 1.692184	WMIN = 1.596841	LMAX = 0.966911	LMIN = 0.936257	
12	WMAX = 1.691265	WMIN = 1.601093	LMAX = 0.966677	LMIN = 0.937926	
	WMAX = 1.690566	WMIN = 1.603586	LMAX = 0.966498	LMIN = 0.938890	
	WMAX = 1.690035	WMIN = 1.604951	LMAX = 0.966362	LMIN = 0.939413	
	WMAX = 1.689630	WMIN = 1.605875	LMAX = 0.966258	LMIN = 0.939766	
	WMAX = 1.689321	WMIN = 1.606499	LMAX = 0.966178	LMIN = 0.940003	
	WMAX = 1.689086	WMIN = 1.606922	LMAX = 0.966117	LMIN = 0.940163	
	WMAX = 1.688905	WMIN = 1.607208	LMAX = 0.966071	LMIN = 0.940272	
	WMAX = 1.688768	WMIN = 1.607358	LMAX = 0.966035	LMIN = 0.940328	
	WMAX = 1.688662	WMIN = 1.607436	LMAX = 0.966008	LMIN = 0.940358	
	WMAX = 1.688582	WMIN = 1.607488	LMAX = 0.965987	LMIN = 0.940378	
	WMAX = 1.688520	WMIN = 1.607524	LMAX = 0.965971	LMIN = 0.940391	
	WMAX = 1.688473	WMIN = 1.607546	LMAX = 0.965959	LMIN = 0.940399	
	WMAX = 1.688437	WMIN = 1.607555	LMAX = 0.965950	LMIN = 0.940403	
	WMAX = 1.688409	WMIN = 1.607560	LMAX = 0.965942	LMIN = 0.940405	
	WMAX = 1.688388	WMIN = 1.607563	LMAX = 0.965937	LMIN = 0.940406	
	WMAX = 1.688372	WMIN = 1.607566	LMAX = 0.965933	LMIN = 0.940407	
	WMAX = 1.688359	WMIN = 1.607567	LMAX = 0.965929	LMIN = 0.940407	
	WMAX = 1.688350	WMIN = 1.607567	LMAX = 0.965927	LMIN = 0.940407	

[a]See pp. 17 and 18.

TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)	IA	I	A(I,1)	A(I,2)	A(I,3)	A(I,4)	I1	I2	I3	I4	K(I)
	1	1	0.	0.	0.	1.00000	5	2	0	6	0.
	1	2	0.	0.	0.	1.00000	1	3	0	7	0.
	1	3	0.	0.	0.	1.00000	2	4	0	8	0.
	1	4	0.	0.	0.	1.00000	3	5	0	9	0.
	1	5	0.	0.	0.	1.00000	4	1	0	10	0.
	2	6	0.25685	0.25685	0.24315	0.24315	10	7	1	11	-0.25685
	2	7	0.25685	0.25685	0.24315	0.24315	6	8	2	12	0.
	2	8	0.25685	0.25685	0.24315	0.24315	7	9	3	13	0.
	2	9	0.25685	0.25685	0.24315	0.24315	8	10	4	14	0.
	2	10	0.25685	0.25685	0.24315	0.24315	9	6	5	15	0.25685
	3	11	0.25685	0.25685	0.24315	0.24315	15	12	6	16	-0.25685
	3	12	0.25685	0.25685	0.24315	0.24315	11	13	7	17	0.
	3	13	0.25685	0.25685	0.24315	0.24315	12	14	8	18	0.
	3	14	0.25685	0.25685	0.24315	0.24315	13	15	9	19	0.
	3	15	0.25685	0.25685	0.24315	0.24315	14	11	10	20	0.25685
	4	16	0.25685	0.25685	0.24315	0.	20	17	11	20	-0.25685
	4	17	0.25685	0.25685	0.24315	0.24315	16	18	12	21	0.
	4	18	0.25685	0.25685	0.24315	0.24315	17	19	13	22	0.
	4	19	0.25685	0.25685	0.24315	0.24315	18	20	14	23	0.
	4	20	0.25685	0.25685	0.24315	0.24315	19	16	15	24	0.25685
	5	21	0.	0.25685	0.24315	0.24315	20	22	17	25	0.
	5	22	0.25685	0.25685	0.24315	0.24315	21	23	18	26	0.
	5	23	0.25685	0.25685	0.24315	0.24315	22	24	19	27	0.
	5	24	0.25685	0.	0.24315	0.24315	23	25	20	28	0.25685
	6	25	0.	0.17878	0.10977	0.10977	24	26	21	29	0.
	6	26	0.25685	0.25685	0.24315	0.24315	25	27	22	30	0.
	6	27	0.25685	0.25685	0.24315	0.24315	26	28	23	31	0.
	6	28	0.19028	0.	0.13779	0.	27	29	24	32	0.67193
	7	29	0.	0.20334	0.13315	0.13315	28	30	25	33	0.
	7	30	0.25685	0.25685	0.24315	0.24315	29	31	26	34	0.
	7	31	0.25513	0.	0.22609	0.	30	32	27	35	0.51878
	8	32	0.	0.07617	0.	0.07381	31	33	28	36	0.
	8	33	0.25685	0.25685	0.24315	0.24315	32	34	29	37	0.
	8	34	0.23245	0.	0.24145	0.	33	35	30	38	0.52610
	9	35	0.	0.16635	0.	0.19622	34	36	31	41	0.
7	9	36	0.25685	0.25685	0.24315	0.24315	35	37	32	42	0.
	9	37	0.17460	0.	0.19424	0.	36	38	33	43	0.63116
	10	38	0.	0.07279	0.	0.08784	37	39	32	44	0.
	10	39	0.16248	0.16248	0.	0.21132	38	40	33	45	0.
	10	40	0.24377	0.24377	0.	0.24283	39	41	34	46	0.
	10	41	0.25685	0.25685	0.24315	0.24315	40	42	35	47	0.
	10	42	0.23445	0.	0.17298	0.17298	41	43	36	48	0.41960
	11	43	0.35764	0.45314	0.	0.09461	48	44	37	49	-0.35764
	11	44	0.48950	0.21593	0.14729	0.14729	43	45	38	50	0.
	11	45	0.25685	0.25685	0.24315	0.24315	44	46	39	51	0.
	11	46	0.25685	0.25685	0.24315	0.24315	45	47	40	52	0.
	11	47	0.25685	0.25685	0.24315	0.24315	46	48	41	53	0.
	11	48	0.23447	0.41952	0.17300	0.17300	47	43	42	54	0.41952
	12	49	0.35764	0.45314	0.09461	0.09461	54	50	43	55	-0.35764
	12	50	0.48950	0.21593	0.14729	0.14729	49	51	44	56	0.
	12	51	0.25685	0.25685	0.24315	0.24315	50	52	45	57	0.
	12	52	0.25685	0.25685	0.24315	0.24315	51	53	46	58	0.
	12	53	0.25685	0.25685	0.24315	0.24315	52	54	47	59	0.
	12	54	0.23447	0.41952	0.17300	0.17300	53	49	48	60	0.41952
	13	55	0.35764	0.45314	0.09461	0.09461	60	56	49	61	-0.35764
	13	56	0.48950	0.21593	0.14729	0.14729	55	57	50	62	0.
	13	57	0.25685	0.25685	0.24315	0.24315	56	58	51	63	0.
	13	58	0.25685	0.25685	0.24315	0.24315	57	59	52	64	0.
	13	59	0.25685	0.25685	0.24315	0.24315	58	60	53	65	0.
	13	60	0.23447	0.41952	0.17300	0.17300	59	55	54	66	0.41952
	14	61	0.35764	0.45314	0.09461	0.09461	66	62	55	67	-0.35764
	14	62	0.48950	0.21593	0.14729	0.14729	61	63	56	68	0.
	14	63	0.25685	0.25685	0.24315	0.24315	62	64	57	69	0.
	14	64	0.25685	0.25685	0.24315	0.24315	63	65	58	70	0.
	14	65	0.25685	0.25685	0.24315	0.24315	64	66	59	71	0.
	14	66	0.23447	0.41952	0.17300	0.17300	65	61	60	72	0.41952
	15	67	0.	0.	1.00000	0.	72	68	61	0	0.48427
	15	68	0.	0.	1.00000	0.	67	69	62	0	0.48427
	15	69	0.	0.	1.00000	0.	68	70	63	0	0.48427
	15	70	0.	0.	1.00000	0.	69	71	64	0	0.48427
	15	71	0.	0.	1.00000	0.	70	72	65	0	0.48427
	15	72	0.	0.	1.00000	0.	71	67	66	0	0.48427

[a]See pp. 17 and 18.

TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)	
8	ERROR = 0.28342844 ERROR = 0.18789074 ERROR = 0.16283273 ERROR = 0.10229385 ERROR = 0.08813190 ERROR = 0.05828751 ERROR = 0.04960943 ERROR = 0.03267638 ERROR = 0.02435843 ERROR = 0.01737347 ERROR = 0.01299492 ERROR = 0.01289449 ERROR = 0.00837459 ERROR = 0.00653413 ERROR = 0.00439080 ERROR = 0.00388455 ERROR = 0.00276583 ERROR = 0.00212220 ERROR = 0.00190944 ERROR = 0.00134461 ERROR = 0.00113655 ERROR = 0.00078486 ERROR = 0.00070114 ERROR = 0.00052264 ERROR = 0.00040352 ERROR = 0.00028567 ERROR = 0.00023039 ERROR = 0.00017443 ERROR = 0.00012948 ERROR = 0.00011159 ERROR = 0.00009041 ERROR = 0.00006004 ERROR = 0.00004838 ERROR = 0.00003868 ERROR = 0.00002921 ERROR = 0.00002357 ERROR = 0.00001960 ERROR = 0.00001372 ERROR = 0.00001073 ERROR = 0.00000825 ERROR = 0.00000640 ERROR = 0.00000500 ERROR = 0.00000434 ERROR = 0.00000317 ERROR = 0.00000226 ERROR = 0.00000176 ERROR = 0.00000139 ERROR = 0.00000113 ERROR = 0.00000092
9	STREAM FUNCTION VALUES 0.02888620 0.22850693 0.43018148 0.63156752 0.83087353 0.02888624 0.22850694 0.43018146 0.63156752 0.83087356 0.02718765 0.22633728 0.43048618 0.63376484 0.83223982 0.02105045 0.21888655 0.43171021 0.64103660 0.83733200 0.19560365 0.43662868 0.66207375 0.85570990 0.12433968 0.45800508 0.71671238 0.92621472 0.19117678 0.55856382 0.82332890 0.07183636 0.45842696 0.76752777 0.21740792 0.50015820 0.80753382 0.06643124 0.24272045 0.45665175 0.68396576 0.90246700 0.47200944 0.55513792 0.74640723 0.95027059 1.15983319 1.36433852 0.95289416 1.03971468 1.23679052 1.43786912 1.64104317 1.84227410 1.43612009 1.52391918 1.72294559 1.92325439 2.12430623 2.32468420 1.92008446 2.00814384 2.20774606 2.40785506 2.60828176 2.80843446 2.40434998 2.49240929 2.69201145 2.89212048 3.09254724 3.29269996

[a]See pp. 17 and 18.

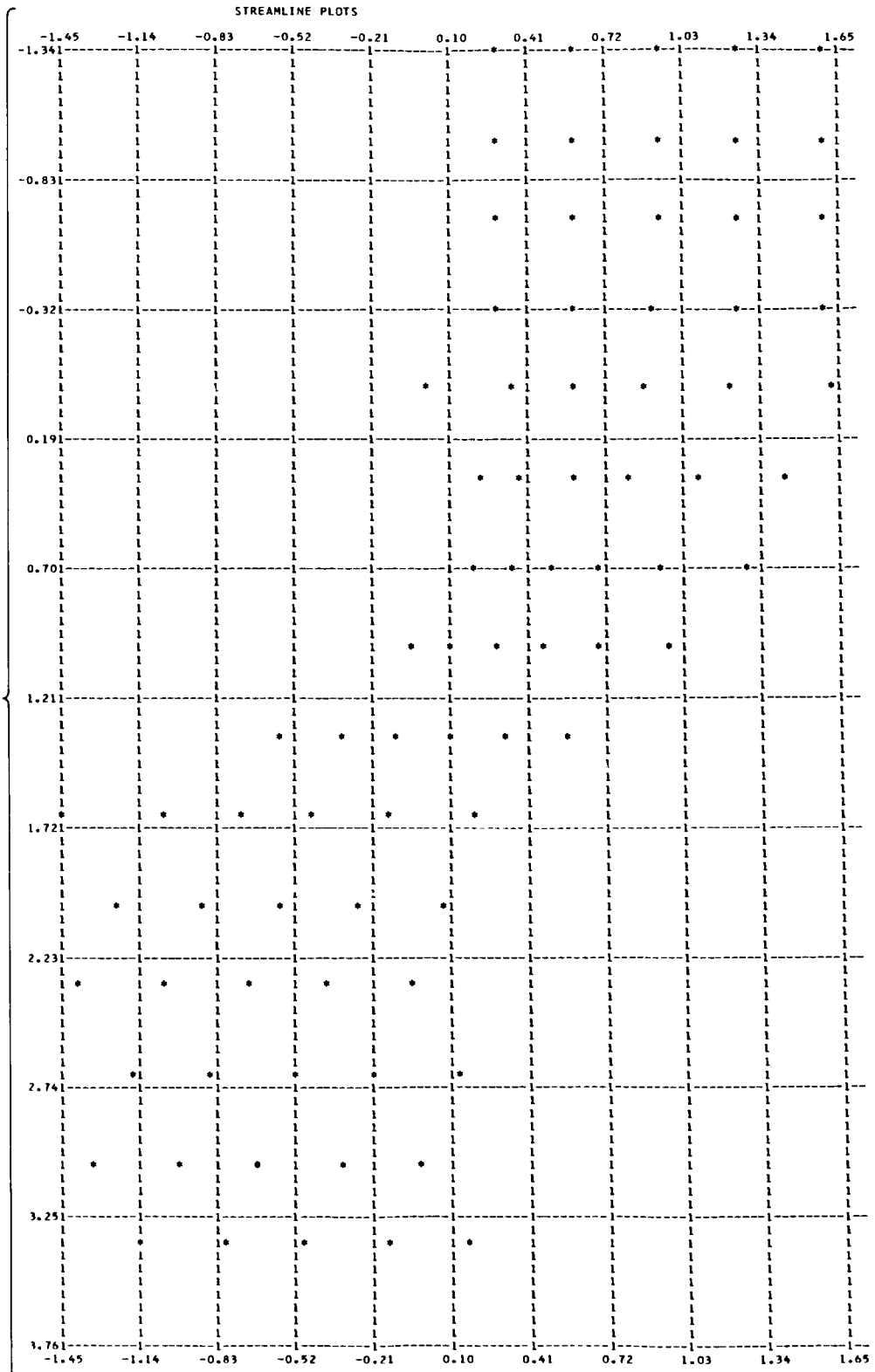
TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)	STREAMLINE COORDINATES						
	Z COORD.	STREAM FN.	X COORD.	STREAM FN.	X COORD.	STREAM FN.	X COORD.
	-1.3432000	0.2000000	0.2802895	0.4000000	0.6046294	0.6000000	0.9287453
		0.8000000	1.2560833	1.0000000	1.5858580		
	-1.0074000	0.2000000	0.2802895	0.4000000	0.6046295	0.6000000	0.9287454
		0.8000000	1.2560832	1.0000000	1.5858580		
	-0.6716000	0.2000000	0.2840282	0.4000000	0.6048338	0.6000000	0.9254065
		0.8000000	1.2533170	1.0000000	1.5878014		
	-0.3358000	0.2000000	0.2966585	0.4000000	0.6052059	0.6000000	0.9146446
		0.8000000	1.2428969	1.0000000	1.5953672		
	0.7450581E-08	0	0	0.2000000	0.3332485	0.4000000	0.6044660
		0.6000000	0.8867970	0.8000000	1.2023373	1.0000000	1.6335999
	0.3358000	0	0.2296385	0.2000000	0.3920730	0.4000000	0.5896215
		0.6000000	0.8229188	0.8000000	1.1040835	1.0000000	1.4310718
3	0.6716000	0	0.2014513	0.2000000	0.3331475	0.4000000	0.4973477
		0.6000000	0.6981699	0.8000000	0.9466724	1.0000000	1.2650239
	1.0074000	0	-0.6240407E-01	0.2000000	0.1072088	0.4000000	0.2748649
		0.6000000	0.4631743	0.8000000	0.6949479	1.0000000	0.9800602
	1.3432000	0	-0.5918410	0.2000000	-0.3477421	0.4000000	-0.1111720
		0.6000000	0.1044712	0.8000000	0.3180047	1.0000000	0.5713504
	1.6790000	0	-1.4509543	0.2000000	-1.0527308	0.4000000	-0.7364154
		0.6000000	-0.4457931	0.8000000	-0.1624246	1.0000000	0.1825543
	2.0148000	0.6000000	-1.2291975	0.8000000	-0.8922368	1.0000000	-0.5757247
		1.2000000	-0.2638732	1.4000000	0.5964373E-01		
	2.3506000	1.0000000	-1.3728456	1.2000000	-1.0407892	1.4000000	-0.7145168
		1.6000000	-0.3927729	1.8000000	-0.6917977E-01		
	2.6864000	1.6000000	-1.1818862	1.8000000	-0.8542062	2.0000000	-0.5287107
		2.1999999	-0.2035202	2.3999999	0.1233529		
	3.0222000	2.0000000	-1.3202116	2.2000000	-0.9928276	2.3999999	-0.6662495
		2.5999999	-0.3402238	2.7999999	-0.1378451E-01		
	3.3580000	2.6000000	-1.1307217	2.8000000	-0.8037332	2.9999999	-0.4775835
		3.1999999	-0.1514143	3.3999999	0.1754820		

[a]See pp. 17 and 18.

Description (a)

TABLE IV. - Continued. SAMPLE OUTPUT



10

[a]See pp. 17 and 18.

TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)	
11	WZ ARRAY 0.6080856 0.6146073 0.6182484 0.6133716 0.6074893 0.6080854 0.6146072 0.6182484 0.6133717 0.6074893 0.6023864 0.6184869 0.6268234 0.6152903 0.6009874 0.5835392 0.6330001 0.6552099 0.6224156 0.5796178 0.6856796 0.7373856 0.6479875 0.5187413 1.2183873 0.8789574 0.7050523 0.6000159 1.4085730 0.9242060 0.6991338 1.1686484 1.1025012 0.8093068 0.8279140 0.9287799 0.8756053 0.4806568 0.6041583 0.6857826 0.7043028 0.5905597 0.5769126 0.5766601 0.6033294 0.6381993 0.6400227 0.6037539 0.6030369 0.6016449 0.6083985 0.6206845 0.6207818 0.6095055 0.6094910 0.6088546 0.6107476 0.6149203 0.6149345 0.6113401 0.6111459 0.6108497 0.6115306 0.6132461 0.6132737 0.6118489 0.6111453 0.6108493 0.6115306 0.6132463 0.6132738 0.6118490 WZU ARRAY 0.5290964 1.3306874 1.6201826 1.1371490 0.8137281 0.4454367 WZL ARRAY 0.3799450 0.5894090 0.5570433 0.6337061 0.7156888 0.4892353 WX ARRAY -0.1970229E-02 -0.7149053E-03 0.4098326E-03 0.1921333E-02 0.5985211E-03 0.2462493E-02 0.8936067E-03 -0.5121660E-03 -0.2401667E-02 -0.7483511E-03 0.7294826E-02 0.1652396E-01 -0.1083348E-02 -0.1194533E-01 -0.9811377E-02 0.3836227E-01 0.1895802E-01 -0.8812166E-02 -0.3441278E-01 -0.1770512E-01 0.1822147 -0.1854434E-01 -0.1033122 -0.1290473 0.9685435E-01 -0.1519258 -0.2284175 -0.2601741 -0.5300829 -0.4631073 -0.4236534 -1.3078349 -0.9612214 -0.7608862 -1.4094831 -1.2103437 -1.1314970 -1.5091154 -1.5558320 -1.4898440 -1.3950833 -1.2715523 -1.4202201 -1.4353746 -1.4680472 -1.4570848 -1.4297067 -1.4239385 -1.4382184 -1.4445926 -1.4528911 -1.4479040 -1.4365135 -1.4288265 -1.4401640 -1.4412398 -1.4446742 -1.4438288 -1.4407349 -1.4403777 -1.4418969 -1.4422873 -1.4428215 -1.4425280 -1.4417496 -1.4412323 -1.4423073 -1.4419945 -1.4415668 -1.4418018 -1.4424249 -1.4428388 ZXU WXU ZXU WXU ZXU WXU ZXU WXU 0.7450581E-08 0.8214765E-01 0.9524691 -1.3076960 1.1908286 -1.4407061 1.3765786 -1.5260978 1.5259748 -1.6103998 1.6352555 -1.5262239 1.6790000 -1.3939518 ZXL WXL ZXL WXL ZXL WXL ZXL WXL 0.6058282 -0.2837112 1.0072824 -0.6084250 1.2835704 -0.9064790 1.5070933 -1.2086117

[a]See pp. 17 and 18.

TABLE IV. - Continued. SAMPLE OUTPUT

Description (a)

4

VELOCITIES AT INTERIOR MESH POINTS

IA= 1	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	0.6081	-0.19	0.6146	-0.07	0.6182	0.04	0.6134	0.18	0.6075	0.06
IA= 2	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	0.6081	0.23	0.6146	0.08	0.6182	-0.05	0.6134	-0.22	0.6075	-0.07
IA= 3	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	0.6024	0.69	0.6187	1.53	0.6268	-0.10	0.6154	-1.11	0.6011	-0.94
IA= 4	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	0.5848	3.76	0.6333	1.72	0.6553	-0.77	0.6234	-3.16	0.5799	-1.75
IA= 5	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	0.7095	14.88	0.7376	-1.44	0.6562	-9.06	0.5346	-13.97		
IA= 6	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.2222	4.55	0.8920	-9.81	0.7411	-17.95	0.6540	-23.44		
IA= 7	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5050	-20.62	1.0337	-26.61	0.8175	-31.21				
IA= 8	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.7539	-48.22	1.4627	-41.08	1.1108	-43.23				
IA= 9	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.6347	-59.57	1.5256	-52.50	1.4307	-52.27				
IA=10	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5838	-72.33	1.6690	-68.78	1.6401	-65.28	1.5628	-63.21	1.4020	-65.09
IA=11	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5329	-67.89	1.5469	-68.11	1.5872	-67.66	1.5907	-66.35	1.5664	-65.88
	1.5466	-67.02								
IA=12	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5595	-67.25	1.5649	-67.39	1.5751	-67.28	1.5753	-66.80	1.5649	-66.63
	1.5534	-66.90								
IA=13	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5638	-67.06	1.5646	-67.10	1.5685	-67.08	1.5693	-66.93	1.5665	-66.89
	1.5647	-67.00								
IA=14	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5661	-67.03	1.5663	-67.05	1.5671	-67.03	1.5675	-66.97	1.5668	-66.96
	1.5657	-67.00								
IA=15	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)	VELOCITY	ANGLE(DEG)
	1.5664	-67.04	1.5660	-67.04	1.5659	-67.01	1.5668	-66.96	1.5674	-66.97
	1.5672	-67.02								

[a]See pp. 17 and 18.

TABLE IV. - Concluded. SAMPLE OUTPUT

Description (a)

SURFACE VELOCITIES BASED ON AXIAL COMPONENTS

UPPER SURFACE				LOWER SURFACE		
Z	VELOCITY	ANGLE(DEG)	WZ	VELOCITY	ANGLE(DEG)	WZ
0.7451E-08	0	90.00	0.5291	0	-90.00	0.3799
0.3358	1.3576	11.43	1.3307	0.6183	-18.19	0.5894
0.6716	1.7447	-21.77	1.6202	0.6710	-35.06	0.5570
1.0074	1.7885	-50.52	1.1371	0.9083	-47.35	0.6337
1.3432	1.6933	-61.28	0.8137	1.2425	-56.74	0.7157
1.6790	0	-90.00	0.4454	0	90.00	0.4892

SURFACE VELOCITIES BASED ON TANGENTIAL COMPONENTS

UPPER SURFACE

Z	VELOCITY	ANGLE(DEG)	WX
0.7451E-08	0.8215E-01	90.00	0.8215E-01
0.9525	1.7950	-46.76	-1.3077
1.1908	1.6894	-58.52	-1.4407
1.3766	1.7314	-61.81	-1.5261
1.5260	1.7191	-69.52	-1.6104
1.6353	1.6002	-72.51	-1.5262
1.6790	1.3940	-90.00	-1.3940

LOWER SURFACE

Z	VELOCITY	ANGLE(DEG)	WX
0.6058	0.5499	-31.06	-0.2837
1.0073	0.8493	-45.75	-0.6084
1.2836	1.1298	-53.36	-0.9065
1.5071	1.4436	-56.85	-1.2086

[a]See pp. 17 and 18.

PROGRAM PROCEDURE

The program is segmented into four main parts, which are the subroutines COEF, SOR, SLAXVL, and TASVEL called by the main program 2DINCP. In addition, there are several other subroutines. All the subroutines and their relation are depicted in figure 9. All information which must be transmitted between the four main subroutines is placed in COMMON. The program can handle up to 2500 mesh points on a computer with a core storage capacity of 32 768.

The first segment of the program is COEF. This subroutine reads all the input cards, calculates the blade coordinates on mesh lines, and calculates the entries of the matrix A and the vector $\underline{k}$ of equation (A6). The subroutine SOR estimates an optimum overrelaxation parameter ω if it is not given as input, calculates an initial solution estimate, and finds the solution to equation (A6) by SOR. Then subroutine SLAXVL calculates the streamline locations and axial velocity components, and plots the streamline locations. Finally, the subroutine TASVEL calculates the tangential velocity components, the velocity magnitudes and direction, and the surface velocities based on axial velocities and on tangential velocities.

Conventions Used in Program

For convenience, a number of conventions are used in naming variables and assigning subscripts. First, several pairs of variables are spelled the same except for one letter, which is U in one case and L in the other. The U signifies the upper surface BC, and L the lower surface CF. Another practice is to use the letters I and O in a similar manner, where I refers to the inlet or the region ABGH, and O refers to the outlet or region CDEF. Thus, ALUO refers to the angle on the upper blade surface near the outlet, or near point C (fig. 8, p. 14).

The variable IA is used as a subscript, or index, associated with vertical mesh lines, from IA = 1 at AH to IA = MX at DE. Similarly IB is used to number horizontal mesh

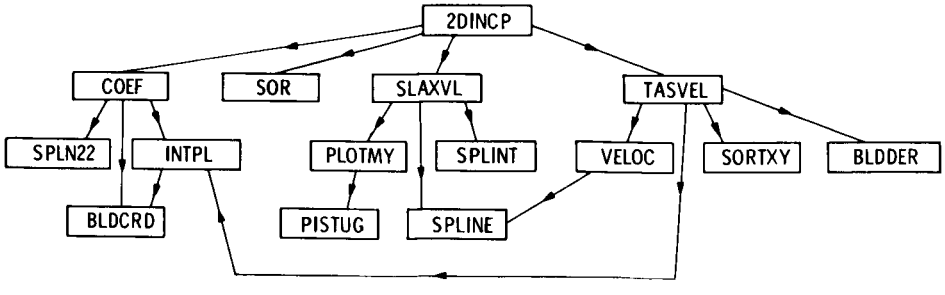


Figure 9. - Logical relation of subroutines.

lines with IB = O along AB. Then IB may be negative along CD. Sometimes IB is used along one vertical mesh line at a time from 1 at the first mesh point to the last mesh point. The variable I is used to number all the mesh points starting with I = 1 at A and proceeding along the vertical mesh lines and moving to the right to the next line after the end of each vertical line and ending with I = NXN at the last mesh point near E. The mesh spacing in the z-direction is labeled HA, and the spacing in the x-direction is HB.

The techniques used in the program and correspondence to the mathematical equations are described briefly. Each subroutine is described separately, first the four segments of the main program, followed by descriptions of each of the remaining subroutines. The various segments of the subroutines are labeled by comment cards, which generally correspond to the headings in the following descriptions.

Subroutine COEF

Input. - The first step is to read all input cards for a particular case. A detailed description of the input required is given in the section Input. All input data are given as the first output.

Calculation of constants. - After all input has been read in, the various constants needed in the program are calculated.

Calculation of mesh coordinates along boundary. - The x-coordinate of boundaries BC and GF at each vertical grid line is calculated by BLDCRD and stored in the arrays XU and XL. BLDCRD requires as input the first and second derivative at each spline point of the cubic spline curves describing the blade surface. These values are calculated by SPLN22. The first and last points of the spline curves are determined by the angle of tangency to the leading- and trailing-edge radii. Therefore, these points need not be specified as input, but are computed by this section of the program.

Calculation of coefficients. - The coefficients of $\underline{u}$ in equations (A1) to (A5), which are the terms of matrix A in equation (A6), are computed at the same time as the constants in these equations, which are the components of $\underline{k}$ in equation (A6), are computed. This computation is performed in three steps, as indicated by comment cards: (1) upstream, (2) between the blades, and (3) downstream. Between the blades it is necessary to compute values for h_3 and h_4 at some mesh points adjacent to the boundary. These values are calculated by INTPL.

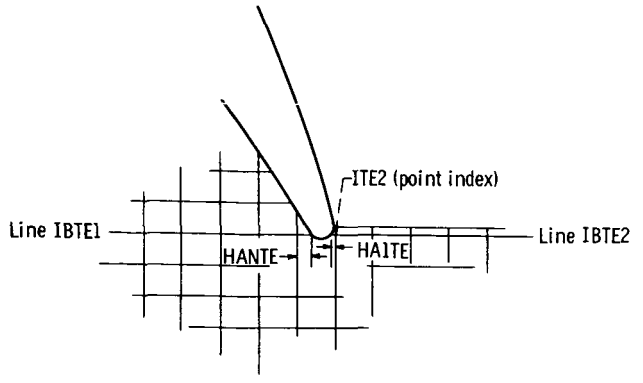


Figure 10. - Special case near trailing edge.

Near the trailing edge, a special situation may arise, as illustrated in figure 10.

Here it should be noted that the blade intersects the mesh line twice between two adjacent mesh points due to the small trailing-edge radius. This situation would not be detected by the program in the normal procedure, which leads to a large error in the velocity calculation at the next to the last vertical grid line on the lower blade surface. Therefore, a special check is made for the two vertical mesh lines involved at statements 113 and 114, and if this situation occurs, the proper values of H3 or H4 are calculated. Also the number of the horizontal mesh line is stored in IBTE2 and IBTE1, so that this fact will be used to calculate the tangential velocity components.

Subroutine SOR

Estimation of value of optimum overrelaxation factor. - If a value of $W \geq 1$ is given as input, it is used for the overrelaxation factor. Otherwise a value is estimated by using equation (B3) to estimate the value of $\rho^2(B) = \rho(L_1)$ and equation (B1) to obtain the corresponding value of W (see appendix B). Equation (A7) is used to calculate u^{m+1} from u^m for equation (B3), with $\omega = 1$ and $\underline{k} = \underline{0}$. To start, $u_i^0 = 1$, for all i. Equation (A7) becomes

$$u_i^{m+1} = -\sum_{j=1}^{i-1} a_{ij} u_j^{m+1} - \sum_{j=i+1}^{n} a_{ij} u_j^m \tag{5}$$

In the program, i is replaced by I directly. For each i, there are only four values of j for which a_{ij} is nonzero, which are the negative values of the coefficients A(I, 1), A(I, 2), A(I, 3), and A(I, 4). The value of j is determined by the index of the proper neighboring point. These indexes are names I1, I2, I3, and I4, and are defined so u_{I1}^m has the coefficient A(I, 1), and similarly for the other coefficients. After the values of the indexes are computed, equation (5) is used to compute $\underline{u}^{m+1}$ from $\underline{u}^m$. Then, the minimum and maximum values of the ratio u_i^{m+1}/u_i^m are calculated and given the names LMIN and LMAX, respectively. After convergence, the optimum value of the overrelaxation factor ω can be calculated from equation (B1), since $\rho^2(B) = LMAX$.

Calculation of initial solution estimate. - The time required to arrive at a solution can be reduced by making an intelligent guess at the solution. A simple way to do this is to use a linear approximation. As input, estimated values of the stream function at A and D are given. It is known that the solution is 1 greater at the points H and E. The stream function is known to be zero at B and C, and 1 at F and G. With this knowledge, bilinear interpolation can be used on the two rectangles ABGH and CDEF, and linear interpolation can be used along vertical mesh lines between the blades to define the initial vector $\underline{u}^0$.

Solution of matrix equation by SOR. - With a value of ω either as input or estimated by the program, equation (A7) can be used iteratively to calculate a sequence $\{\underline{u}^m\}$ that will converge rapidly to a solution of equation (A6). The indexes i and j and the correspondence of a_{ij} and u_j^m to the program variables is the same as described previously for estimating the optimum overrelaxation factor. During each iteration the maximum of the change of the stream-function value is calculated. When this value is reduced below TOLER given as input, the iteration is stopped, and the current estimate of the stream function is accepted as the solution.

Subroutine SLAXVL

Calculation of streamline locations and axial velocity components. - Along most vertical mesh lines, the stream function is a one to one function of the distance in the x-direction. Therefore, the x-dimension is considered to be a function of the value of the stream function, and the value of x at a given value of the stream function can be obtained by interpolation. The method of interpolating uses a cubic spline curve, and the interpolation is performed by subroutine SPLINT. At the same time, $\partial u/\partial x = V_z$ is computed along the same mesh line, estimating the derivative by use of the cubic spline. This calculation at each mesh point is done by SPLINE. The axial velocity at unknown mesh points is stored in the array WZ, and the axial velocity along the blade surfaces is stored in WZU for the upper surface of the blade and in WZL for the lower surface. These calculations are performed in three sections, as noted by comment cards: (1) upstream, (2) between the blades, and (3) downstream.

Plotting of streamlines. - The streamlines can be plotted to give a rough idea of their locations. These locations are particularly helpful in quickly disclosing any errors of input. The plotting printout is done by PLOTMY, which, with the necessary further subroutine PISTUG, is described completely with FORTRAN II listing in references 7 and 8. These programs are available as SHARE No. SDA 3034. The plotting can be omitted by removing statements following statement 420 up to and including statement 470.

Subroutine TASVEL

Calculation of tangential velocity components. - The tangential velocity component is calculated from $\partial u/\partial z = -V_x$ by considering each horizontal mesh line. The fact that the various horizontal mesh lines start and end at various places complicates this process. To simplify the procedure, upstream and downstream ends of each mesh line are considered separately. At the upstream end, there are three possibilities: (1) the line starts

at AH(IA = 1), (2) the line starts on lower surface of blade, or (3) the line starts on the upper surface of the blade. Similarly, at the downstream end there are three possibilities: (1) the line ends at DE(IA = MX), (2) the line ends on the lower surface, or (3) the line ends on the upper surface. For convenience, case numbers are assigned to the various possibilities as follows:

Case	Starts	Ends
1	AH	Upper surface
2	AH	DE
3	AH	Lower surface
4	Lower surface	DE or lower surface
5	Upper surface	DE or lower surface

As mentioned in the description of COEF, a special situation often arises where a horizontal mesh line is intersected twice between two adjacent mesh points, as illustrated in figure 10 (p. 29). When this occurs, the index of the horizontal mesh line is stored in IBTE2 and IBTE1. Under cases 2, 4, or 5, if IB = IBTE2, the special case arises, and previously calculated information is used for this line to the left of the trailing edge. After all other tangential velocities have been calculated, tangential velocities are calculated for the remainder of this line (IB = IBTE1) to the right of the trailing edge.

TASVEL calculates the necessary information about the two end points of each horizontal mesh line by using INTPL to calculate the mesh spacing at the end points. Then VELOC calculates $\partial u/\partial z$ at each point along the line by using SPLINE to calculate the actual derivatives. The tangential velocities at unknown mesh points are stored in the array WX, and the tangential velocities along the upper surface are stored in WXU, with the corresponding z-coordinates stored in ZXU. The corresponding information for the lower surface is stored in WXL and ZXL. After all values for velocities are computed, the signs are changed, since $V_x = -\partial u/\partial z$. The values of WXL and ZXL are rearranged in increasing order of ZXL by SORTXY.

Calculation of velocities and angles at interior points. - At each interior point, the velocity magnitude is calculated by $V = \sqrt{V_x^2 + V_z^2}$, and the angle θ is calculated by $\tan \theta = V_x/V_z$.

Calculation of surface velocity based on axial components. - The surface velocity at each vertical mesh line is calculated by

$$V = \frac{V_z}{\cos \theta} = V_z \sqrt{1 + \left(\frac{dx}{dz}\right)^2}$$

The slope dx/dz of the blade surface at each vertical mesh line is computed by BLDCRD,

at the same time the blade coordinates XU and XL are computed, and is stored in DXDZU and DXDZL for the upper and lower blade surfaces. The surface velocity based on axial components is more accurate at small angles from the horizontal and would not be expected to be accurate at angles over 60° from the horizontal.

Calculation of surface velocities based on tangential velocity components. - The surface velocity at each horizontal mesh lines is calculated by

$$V = \frac{V_x}{\sin \theta} = V_x \sqrt{1 + \frac{1}{\left(\frac{dx}{dz}\right)^2}}$$

The slope dx/dz of the blade surface at each horizontal mesh line is computed by BLDDER and is stored in DXDZU and DXDZL. The surface velocity based on tangential components is more accurate at large angles from the horizontal and would not be expected to be accurate at angles less than 30° from the horizontal.

Internal Variables for COEF, SOR, SLAXVL, and TASVEL

A	array of coefficients of u which are elements of matrix A in equation (A6)
AAA	array used for temporary storage
AII	a_0 in equation (A1)
B	temporary storage
CASE	number (integer) of case in calculating tangential velocity components
CHANGE	change in value of stream function at a particular point when using SOR iteration
DELINT	increment of stream function for which streamline locations are to be calculated
DXDZL	array of values of slope of lower blade surface at each vertical mesh line, and later at each horizontal mesh line
DXDZU	same as DXDZL, but for upper blade surface
EML	array of second derivatives of spline curve at each spline point for lower blade surface, calculated by SPLN22
EMU	same as EML, but for upper blade surface
FIRST	value (integer) of I at lowest mesh point for given vertical mesh line

H1	h_1 (see fig. 13 in appendix A)
H2	h_2 (see fig. 13 in appendix A)
H3	h_3 (see fig. 13 in appendix A)
H4	h_4 (see fig. 13 in appendix A)
HA	basic mesh space in axial (z) direction
HB	basic mesh space in blade-to-blade (x) direction
HA1	length of first mesh space along horizontal mesh line
HA1TE	HA1 for special case shown in figure 10 for line segment to right of trailing edge
HAN	length of last mesh space along horizontal mesh line
HANTE	HAN for special case shown in figure 10 for line segment to left of trailing edge
HL	distance between EF on boundary and first mesh line below (fig. 3)
HU	distance between CD on boundary and first mesh line above (fig. 3)
I	index of mesh point
I1	index of mesh point located at 1 in figure 13 with I at 0
I2	index of mesh point located at 2 in figure 13 with I at 0
I3	index of mesh point located at 3 in figure 13 with I at 0
I4	index of mesh point located at 4 in figure 13 with I at 0
IA	index of mesh line in axial (z) direction
IB	index of mesh line in blade-to-blade (x) direction
IBTE1	index of special mesh line shown in figure 10
IBTE2	index of special mesh line shown in figure 10
IA1	index of first mesh point along horizontal mesh line
IAN	index of last mesh point along horizontal mesh line
IL	array of indexes of highest mesh point for each vertical mesh line
ITE2	index of mesh point indicated in figure 10
IU	array of indexes of lowest mesh point for each vertical mesh line
J	temporary index
J1	temporary index

JB	temporary index
JL	number of points where horizontal mesh line intersects lower blade surface
JU	number of points where horizontal mesh line intersects upper blade surface
JUM1	JU-1
K	array (real) of constants that is vector $\underline{k}$ in equation (A6)
K1, K2, K3, K4, K5	code variables (real) used in determining values of coefficients A(I, J) and constants K(I)
KK1	code to specify whether first point of horizontal mesh line is on AH(KK1 = 0) or upper blade surface (KK1 = 0) or lower blade surface (KK1 = 1)
KN	same as KK1, but for last point
KKK	array containing information used in plotting subroutine PLOTMY
LAST	value of I at highest mesh point for given vertical mesh line
LMAX	upper bound (real) for $\rho(L_1)$ from equation (B2)
LMIN	lower bound (real) for $\rho(L_1)$ from equation (B2)
MXBIM1	MXBI - 1
MXBIP1	MXBI + 1
MXBOM1	MXBO - 1
MXBOP1	MXBO + 1
NBB	number of mesh points along vertical mesh line
NBBO	number of mesh lines above mesh line AB for first mesh line below EF (may be negative)
NBUO	number of mesh lines above mesh line AB for line CD (usually negative, unless STGR is positive)
NCH	number of vertical mesh lines in length of blade
NL	array of number of mesh points on vertical mesh line above line AB (may be negative)
NSP	number of mesh points plus boundary points along vertical mesh line
NU	array; on vertical mesh line IA, the mesh point nearest the upper blade surface is NU(IA) mesh points above line AB (NU(IA) may be negative)

P	array of input information for plotting subroutine PLOTMY
RATIO	value of u_i^{m+1}/u_i^m for use in equations (B2) and (B3)
SL	array of streamline coordinates for input data to the plotting subroutine PLOTMY
SLLPE	array of slopes of spline curve at each spline point for lower blade surface, calculated by SPLN22
SLUPE	same as SLLPE, but for upper blade surface
SRW	code (integer) variable that will cause certain subroutines to write out data useful for debugging: SRW = 18 SPLN22 will write input and output data = 19 BLDCRD will write out blade coordinates and slopes at each mesh line = 19 INTPL will write out pertinent data for each iteration = 16 SPLINT will write input and output data = 13 SPLINE will write input and output data = 20 BLDDER will write out z-coordinates and slopes
TANTH	tan θ at unknown mesh points
TANTHL	tan θ along lower blade surface
TANTHU	tan θ along upper blade surface
THETA	streamline angle θ at unknown mesh points
THETAL	streamline angle θ along lower blade surface
THETAU	streamline angle θ along upper blade surface
U	array of estimated values of stream function or of eigenvector associated with spectral radius of L_1 and $\rho(L_1)$, as estimated by power method
UINT	array of values of stream function for which it is desired to obtain interpolated values of x-coordinate
UNEW	new value of eigenvector estimate at single point, as calculated by equation (5) (p. 30)
UNI	$(\partial u/\partial n)_{in}$ (eqs. (3) and (A2))
UNO	$(\partial u/\partial n)_{out}$ (eqs. (3) and (A3))
USP	array of values of stream function along vertical mesh line, including boundary points
V	array of velocities at unknown mesh points

WL	array of velocities along lower blade surface
WMAX	upper bound for optimum ω from equations (B1) and (B2)
WMIN	lower bound for optimum ω from equations (B1) and (B2)
WU	array of velocities along upper blade surface
WX	array of tangential velocity components at unknown mesh points
WXL	array of tangential components of velocities where horizontal mesh lines intersect lower blade surface
WXU	array of tangential velocity components where horizontal mesh lines intersect upper blade surface
WZ	array of axial velocity components at unknown mesh points
WZL	array of axial velocity components where vertical mesh lines intersect lower blade surface
WZU	array of axial velocity components where vertical mesh lines intersect upper blade surface
X1	value of x for which INTPL is to compute H3 or H4
XBB	array of x-coordinates associated with array USP
XINT	array of interpolated x-coordinates calculated by SPLINT and corresponding to array UINT
XL	array of x-coordinates of lower surface of blade at each vertical mesh line
XU	array of x-coordinates of upper surface of blade at each vertical mesh line
ZINT	argument for BLDCRD, not used in main program
ZPL	array of z-coordinates of vertical mesh lines
ZXL	array of z-coordinates of intersections of horizontal mesh lines with lower blade surface
ZXU	array of z-coordinates of intersections of horizontal mesh lines with upper blade surface

```
$IBFTC 2DINCP

   10 CALL COEF(NXN)
      IF(NXN.GT.2500) GO TO 10
      CALL SOR
      CALL SLAXVL
      CALLTASVEL
      GO TO 10
      END


$IBFTC COEF

      SUBROUTINE COEF(NXN1)
C
C    IA IS AXIAL INDEX
C    IB IS BLADE-TO-BLADE INDEX
C    I IS OVERALL INDEX
C    HA IS BASIC AXIAL INCREMENT
C    HB IS BASIC BLADE-TO-BLADE INCREMENT
C
      REAL K,K1,K2,K3,K4,K5,LMAX,LMIN
      INTEGER SRW,FIRST,CASE,BLDATA,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,SURVEL
      COMMON SRW,PITCH,CHORD,STGR,THETAI,THETAO,DTLR,RI,ALUI,ALLI,
     1   RO,ALUO,ALLO,MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT,
     2   BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,SURVEL,
     3   ZU(50),XSPU(50),ZL(50),XSPL(50),W,WR,TOLER,BDA,BDD,
     4   U(2500),A(2500,4),K(2500),
     5   DXDZU(100),DXDZL(100),SLUPE(50),EMU(50),SLLPE(50),EML(50),
     6   XU(100),XL(100),NU(100),NL(100),UINT(11),XINT(11),
     7   HA,HB,NXN,MXBIM1,MXBOP1,JU,JL,HU,HL,ID,NBBO,NBUO,NCH,
     8   IBTE1,IBTE2,ITE2,HA1TE,HANTE
      DIMENSION WZ(2500),WX(2500),V(2500),THETA(2500),SL(1100)
      DIMENSION WZU(100),WZL(100),WXU(100),WXL(100),ZXU(100),ZXL(100),
     1   THETAU(100),THETAL(100),WU(100),WL(100),
     2   USP(100),ZPL(100),IU(100),IL(100),AAA(100)
      EQUIVALENCE (A(1,1),WZ(1)),(A(1,2),WX(1)),(A(1,3),V(1)),
     1   (A(1,4),THETA(1)),(K(1501),SL(1))
      EQUIVALENCE (K(1),WZU(1)),(K(101),WZL(1)),(K(201),WXU(1)),
     1   (K(301),WXL(1)),(K(401),ZXU(1)),(K(501),ZXL(1)),
     2   (K(601),THETAU(1)),(K(701),THETAL(1)),(K(801),WU(1)),
     3   (K(901),WL(1)),(K(1001),USP(1)),(K(1101),ZPL(1)),
     4   (K(1201),IU(1)),(K(1301),IL(1)),(K(1401),AAA(1))
   10 WRITE (6,999)
      READ (5,1010) PITCH,CHORD,STGR,THETAI,THETAO,DTLR
      WRITE(6,1020) PITCH,CHORD,STGR,THETAI,THETAO,DTLR
      READ (5,1010) RI,ALUI,ALLI,RO,ALUO,ALLO
      WRITE(6,1020) RI,ALUI,ALLI,RO,ALUO,ALLO
      READ (5,1000)MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT
      WRITE(6,1000)MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT
      READ (5,1010) (  ZU(IA),IA=1,NUSP)
      WRITE(6,1020) (  ZU(IA),IA=1,NUSP)
      READ (5,1010) (XSPU(IA),IA=1,NUSP)
      WRITE(6,1020) (XSPU(IA),IA=1,NUSP)
      READ (5,1010) (  ZL(IA),IA=1,NLSP)
      WRITE(6,1020) (  ZL(IA),IA=1,NLSP)
      READ (5,1010) (XSPL(IA),IA=1,NLSP)
      WRITE(6,1020) (XSPL(IA),IA=1,NLSP)
      READ (5,1000) BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,
     1   SURVEL
      WRITE(6,1000) BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,
     1   SURVEL
      READ (5,1010) W,WR,TOLER,BDA,BDD
```

```
      WRITE(6,1020) W,WR,TOLER,BDA,BDD
C
C     END OF INPUT
C
      HA = CHORD/FLOAT(MXBO-MXBI)
      HB = PITCH/FLOAT(NBBI)
      A = (PITCH+STGR)/HB
      B = STGR/HB
      NBBO = A-SIGN(DTLR,A)
      IF(A.LT.DTLR) NBBO = NBBO-1
      NBUO = B-SIGN(DTLR,B)
      IF(B.LT.-DTLR) NBUO = NBUO-1
      HU = FLOAT(NBUO+1)*HB-STGR
      HL = PITCH+STGR-FLOAT(NBBO)*HB
      THETAI = THETAI/57.29577
      THETAO = THETAO/57.29577
      UNI =  SIN(THETAI)/COS(THETAI)/PITCH
      UNO = -SIN(THETAO)/COS(THETAO)/PITCH
      ALUI = ALUI/57.29577
      ALUO = ALUO/57.29577
      ALLI = ALLI/57.29577
      ALLO = ALLO/57.29577
      ALUI = SIN(ALUI)/COS(ALUI)
      ALUO = SIN(ALUO)/COS(ALUO)
      ALLI = SIN(ALLI)/COS(ALLI)
      ALLO = SIN(ALLO)/COS(ALLO)
      NCH = MXBO-MXBI+1
      MXBIM1 = MXBI-1
      MXBOP1 = MXBO+1
      IBTE1 = 1000
      IBTE2 = 1000
C
C     CALCULATE MESH COORDINATES ON BOUNDARY
C
      WRITE (6,999)
      ZU(1) = RI*(1.-ALUI/SQRT(1.+ALUI**2))
      ZU(NUSP) = CHORD-RO*(1.+ALUO/SQRT(1.+ALUO**2))
      ZL(1) = RI*(1.+ALLI/SQRT(1.+ALLI**2))
      ZL(NLSP) = CHORD-RO*(1.-ALLO/SQRT(1.+ALLO**2))
      XSPU(1) = SQRT(ZU(1)*(2.*RI-ZU(1)))
      XSPL(1) =-SQRT(ZL(1)*(2.*RI-ZL(1)))+PITCH
      XSPU(NUSP) = SQRT((CHORD-ZU(NUSP))*(2.*RO-CHORD+ZU(NUSP)))+STGR
      XSPL(NLSP) = -SQRT((CHORD-ZL(NLSP))*(2.*RO-CHORD+ZL(NLSP)))+STGR
     1    +PITCH
      IF(BLDATA.EQ.1) SRW=18
      CALL SPLN22(ZU,XSPU,ALUI,ALUO,NUSP,SLUPE,EMU)
      CALL SPLN22(ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML)
      IF(BLDATA.EQ.1) SRW=19
      CALL BLDCRD (ZU,XSPU,SLUPE,EMU,NUSP,RI,ALUI,RO,ALUO,CHORD,STGR,
     1    PITCH, 1.,XU,NCH,ZINT,MXBI,DXDZU)
      CALL BLDCRD (ZL,XSPL,SLLPE,EML,NLSP,RI,ALLI,RO,ALLO,CHORD,STGR,
     1    PITCH,-1.,XL,NCH,ZINT,MXBI,DXDZL)
      SRW = 0
C
C     CALCULATE UPSTREAM COEFFICIENTS
C
```

```
   90 CONTINUE
      I = 0
      DO 100 IA = 1,MXBIM1
      NL(IA) = NBBI-1
      NU(IA) = 0
      NBB = NL(IA)-NU(IA)+1
      XU(IA) = 0.
      XL(IA) = PITCH
      DO 100 IB=1,NBB
      I = I+1
      AII = 2.*(HA**2+HB**2)
      A(I,1) = HA**2/AII
      A(I,2) = A(I,1)
      A(I,3) = HB**2/AII
      A(I,4) = A(I,3)
      K(I) = 0.
      IF(IB.EQ.1) K(I) = -A(I,1)+K(I)
      IF(IB.EQ.NBB) K(I) = A(I,2)+K(I)
      IF((IA.EQ.MXBIM1).AND.(IB .EQ.1)) A(I,4) = 0.
      IF(IA.NE.1) GO TO 100
      A(I,1) = 0.
      A(I,2) = 0.
      A(I,3) = 0.
      A(I,4) = 1.
      K(I) = HA*UNI
  100 CONTINUE
C
C   CALCULATE COEFFICIENTS BETWEEN BLADES
C
      DO 110 IA=MXBI,MXBO
      NU(IA) = INT((XU(IA)+DTLR)/HB)
      IF(XU(IA).GT.-DTLR) NU(IA)=NU(IA)+1
      NL(IA) = INT((XL(IA)-DTLR)/HB)
      IF(XL(IA).LT.DTLR) NL(IA) = NL(IA)-1
  110 CONTINUE
      NL(MXBOP1) = NBBO
      NU(MXBOP1) = NBUO
      DO 120 IA = MXBI,MXBO
      X1 = FLOAT(NU(IA))*HB
      NBB = NL(IA)-NU(IA)+1
      K5 = 0.
      DO 120 IB=1,NBB
      I = I+1
      IF(I.GT.2500) GO TO 120
      K1 = 0.
      K2 = 0.
      K3 = 0.
      K4 = 0.
      H3 = HA
      H4 = HA
      IF((NU(IA)+IB).LE.NU(IA-1)) CALL INTPL(XU,IA,X1,H3,0,K3,ZU,XSPU,
     1   ALUI,ALUO,NUSP,SLUPE,EMU,RI,RO,CHORD,STGR,PITCH, 1.,HA,HB,
     2   MXBI,MXBO)
      IF((NU(IA)+IB).LE.NU(IA+1)) CALL INTPL(XU,IA,X1,H4,1,K4,ZU,XSPU,
     1   ALUI,ALUO,NUSP,SLUPE,EMU,RI,RO,CHORD,STGR,PITCH, 1.,HA,HB,
     2   MXBI,MXBO)
```

```
      IF((NU(IA)+IB).GT.(NL(IA-1)+1))  CALL INTPL(XL,IA,X1,H3,0,K3,ZL,
     1   XSPL,ALLI,ALLO,NLSP,SLLPE,EML,RI,RO,CHORD,STGR,PITCH,-1.,HA,HB,
     2   MXBI,MXBO)
      IF((NU(IA)+IB).GT.(NL(IA+1)+1)) CALL INTPL(XL,IA,X1,H4,1,K4,ZL,
     1   XSPL,ALLI,ALLO,NLSP,SLLPE,EML,RI,RO,CHORD,STGR,PITCH,-1.,HA,HB,
     2   MXBI,MXBO)
      IF(IA-MXBO+1) 115,113,114
C   SPECIAL CALCULATION OF COEFFICIENTS AT TRAILING EDGE OF LOWER SURFACE OF
C      UPPER BLADE
  113 IF(X1.LT.PITCH+STGR-RO) GO TO 115
      IF(X1.GT.PITCH+STGR) GO TO 115
      HAMRO = HA-RO
      XL(MXBO) = PITCH+STGR-RO
      CALL INTPL(XL,IA,X1,H4,1,K4,ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML,
     1   RI,RO,CHORD,STGR,PITCH,-1.,HAMRO,HB,MXBI,MXBO)
      HANTE = H4
      IBTE2 = NU(IA)+IB-1
      XL(MXBO) = PITCH+STGR
      GO TO 115
  114 IF(X1.LT.PITCH+STGR-RO) GO TO 115
      IF(IBTE2.EQ.1000) GO TO 115
      B = XL(MXBO-1)
      XL(MXBO-1) = PITCH+STGR-RO
      CALL INTPL(XL,IA,X1,H3,0,K3,ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML,
     1   RI,RO,CHORD,STGR,PITCH,-1.,RO,HB,MXBI,MXBO)
      HA1TE = H3
      IBTE1 = NU(IA)+IB-1
      ITE2 = I
      XL(MXBO-1) = B
  115 IF(IB.EQ.1) K1 = 1.
      IF(IB.EQ.NBB) K2 = 1.
      H1 = HB-K1*(XU(IA)-X1+HB)
      H2 = HB-K2*(X1-XL(IA)+HB)
      AII = (H3+H4)*(1./H1+1./H2)+(H1+H2)*(1./H3+1./H4)
      A(I,1) = (H3+H4)/H1/AII
      A(I,2) = (H3+H4)/H2/AII
      A(I,3) = (H1+H2)/H3/AII
      A(I,4) = (H1+H2)/H4/AII
      IF(K3.LT..5.AND.K4.LT..5) K5 = 1.
      K(I) = K5*(K2*A(I,2)+K3*A(I,3)+K4*A(I,4))
      IF(K1.GT..5) A(I,1) = 0.
      IF(K2.GT..5) A(I,2) = 0.
      IF(K3.GT..5) A(I,3) = 0.
      IF(K4.GT..5) A(I,4) = 0.
      X1 = X1+HB
  120 CONTINUE
C
C   CALCULATE DOWNSTREAM COEFFICIENTS
C
      DO 170 IA=MXBOP1,MX
      NU(IA) = NBUO
      NL(IA) = NBBO
      XU(IA) = STGR
      XL(IA) = PITCH+STGR
      I = I+1
      IF(I.GT.2500) GO TO 140
```

```
      AII = (HA+HA)*(1./HL+1./HU)+(HL+HU)*2./HA
      A(I,1) = (HA+HA)/HL/AII
      A(I,2) = (HA+HA)/HU/AII
      A(I,3) = (HL+HU)/HA/AII
      A(I,4) = A(I,3)
      K(I) = -A(I,1)
      IF(IA.EQ.MXBOP1) A(I,3) = 0.
      IF(IA.NE.MX) GO TO 140
      A(I,1) = 0.
      A(I,2) = 0.
      A(I,4) = 0.
      A(I,3) = 1.
      K(I) = HA*UNO
  140 I = I+1
      IF(I.GT.2500) GO TO 150
      AII = (HA+HA)*(1./HB+1./HU)+(HB+HU)*2./HA
      A(I,1) = (HA+HA)/HU/AII
      A(I,2) = (HA+HA)/HB/AII
      A(I,3) = (HB+HU)/HA/AII
      A(I,4) = A(I,3)
      K(I) = 0.
      IF(IA.NE.MX) GO TO 150
      A(I,1) = 0.
      A(I,2) = 0.
      A(I,4) = 0.
      A(I,3) = 1.
      K(I) = K(I-1)
  150 NBB = NL(IA)-NU(IA)
      DO 160 IB=3,NBB
      I = I+1
      IF(I.GT.2500) GO TO 160
      AII = 2.*(HA**2+HB**2)
      A(I,1) = HA**2/AII
      A(I,2) = A(I,1)
      A(I,3) = HB**2/AII
      A(I,4) = A(I,3)
      K(I) = 0.
      IF(IA.NE.MX) GO TO 160
      A(I,1) = 0.
      A(I,2) = 0.
      A(I,4) = 0.
      A(I,3) = 1.
      K(I) = K(I-1)
  160 CONTINUE
      I = I+1
      IF(I.GT.2500) GO TO 170
      AII = (HA+HA)*(1./HB+1./HL)+(HB+HL)*2./HA
      A(I,1) = (HA+HA)/HB/AII
      A(I,2) = (HA+HA)/HL/AII
      A(I,3) = (HB+HL)/HA/AII
      A(I,4) = A(I,3)
      K(I) = A(I,2)
      IF(IA.NE.MX) GO TO 170
      A(I,1) = 0.
      A(I,2) = 0.
      A(I,4) = 0.
```

```
      A(I,3) = 1.
      K(I) = K(I-1)
  170 CONTINUE
      NXN = I
      WRITE (6,1420) NXN
      IF(NXN.GT.2500) WRITE(6,1430)
      NXN1 = NXN
      IF(NULAKI.EQ.0) RETURN
      WRITE (6,1410)  (NU(IA),NL(IA) ,IA=1,MX)
      RETURN
  999 FORMAT (1H1)
 1000 FORMAT (16I5)
 1010 FORMAT (8F10.5)
 1020 FORMAT (1X,8G16.7)
 1410 FORMAT (18H LIST OF NU AND NL / (2I10))
 1420 FORMAT (1H ,5X,32HNUMBER OF INTERIOR MESH POINTS =,I5)
 1430 FORMAT(76HKTHE NUMBER OF UNKNOWN MESH POINTS EXCEEDS 2500, A COARS
     1ER GRID MUST BE USED)
      END
```

```
$IBFTC SOR

      SUBROUTINE SOR
      REAL K,K1,K2,K3,K4,K5,LMAX,LMIN
      INTEGER SRW,FIRST,CASE,BLDATA,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,SURVEL
      COMMON SRW,PITCH,CHORD,STGR,THETAI,THETAO,DTLR,RI,ALUI,ALLI,
     1   RO,ALUO,ALLO,MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT,
     2   BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,SURVEL,
     3   ZU(50),XSPU(50),ZL(50),XSPL(50),W,WR,TOLER,BDA,BDD,
     4   U(2500),A(2500,4),K(2500),
     5   DXDZU(100),DXDZL(100),SLUPE(50),EMU(50),SLLPE(50),EML(50),
     6   XU(100),XL(100),NU(100),NL(100),UINT(11),XINT(11),
     7   HA,HB,NXN,MXBIM1,MXBOP1,JU,JL,HU,HL,ID,NBBO,NBUO,NCH,
     8   IBTE1,IBTE2,ITE2,HA1TE,HANTE
      DIMENSION WZ(2500),WX(2500),V(2500),THETA(2500),SL(1100)
      DIMENSION WZU(100),WZL(100),WXU(100),WXL(100),ZXU(100),ZXL(100),
     1   THETAU(100),THETAL(100),WU(100),WL(100),
     2   USP(100),ZPL(100),IU(100),IL(100),AAA(100)
      EQUIVALENCE (A(1,1),WZ(1)),(A(1,2),WX(1)),(A(1,3),V(1)),
     1   (A(1,4),THETA(1)),(K(1501),SL(1))
      EQUIVALENCE (K(1),WZU(1)),(K(101),WZL(1)),(K(201),WXU(1)),
     1   (K(301),WXL(1)),(K(401),ZXU(1)),(K(501),ZXL(1)),
     2   (K(601),THETAU(1)),(K(701),THETAL(1)),(K(801),WU(1)),
     3   (K(901),WL(1)),(K(1001),USP(1)),(K(1101),ZPL(1)),
     4   (K(1201),IU(1)),(K(1301),IL(1)),(K(1401),AAA(1))
C
C   ESTIMATE WBEST
C
      IF(W.GE.1.) GO TO 225
  190 DO 200 I=1,NXN
  200 U(I) = 1.
      WMAX = 2.
  210 I = 0
      WMAX1 = WMAX
      LMAX = 0.
      LMIN = 1.
      DO 220 IA=1,MX
      NBB = NL(IA)-NU(IA)+1
      DO 220 IB =1,NBB
      I = I+1
      I1 = I-1
      I2 = I+1
      IF(((IA.LT.MXBI).OR.(IA.GT.MXBO)).AND.(IB.EQ.1)) I1 = I1+NBB
      IF(((IA.LT.MXBI).OR.(IA.GT.MXBO)).AND.(IB.EQ.NBB)) I2 = I2-NBB
      I3 = I-NL(IA-1)+NU(IA)-1
      I4 = I+NL(IA)-NU(IA+1)+1
      UNEW = A(I,1)*U(I1)+A(I,2)*U(I2)+A(I,3)*U(I3)+A(I,4)*U(I4)
      RATIO = UNEW/U(I)
      LMAX = AMAX1(RATIO,LMAX)
      LMIN = AMIN1(RATIO,LMIN)
  215 U(I) = UNEW
  220 CONTINUE
```

```
      WMAX = 2./(1.+SQRT(ABS(1.-LMAX)))
      WMIN = 2./(1.+SQRT(1.-LMIN))
      WRITE (6,1500) WMAX,WMIN,LMAX,LMIN
      IF(((WMAX1-WMAX).GT.WR).OR.(WMAX.GT.(2.-100.*WR))) GO TO 210
      W = WMAX
C
C   CALCULATE INITIAL SOLUTION ESTIMATE
C
  225 I = 0
      DO 230 IA=1,MXBIM1
      NBB = NL(IA)-NU(IA)+1
      X1 = FLOAT(MXBI-IA)/FLOAT(MXBI)*BDA
      DO 230 IB=1,NBB
      I = I+1
      U(I) = X1+FLOAT(IB-1)/FLOAT(NBB)
  230 CONTINUE
      DO 240 IA=MXBI,MXBO
      NBB = NL(IA)-NU(IA)+1
      DO 240 IB=1,NBB
      I = I+1
      J = NU(IA)+IB-1
      U(I) = (HB*FLOAT(J)-XU(IA))/(XL(IA)-XU(IA))
  240 CONTINUE
      DO 250 IA=MXBOP1,MX
      NBB = NL(IA)-NU(IA)+1
      X1 = FLOAT(IA-MXBO)/FLOAT(MX-MXBO)*BDD
      DO 250 IB=1,NBB
      I = I+1
      U(I) = X1+FLOAT(IB-1)/FLOAT(NBB)
  250 CONTINUE
C
C   SOLVE MATRIX EQUATION BY SOR
C
      IF(NULAKI.NE.0) WRITE (6,1450)
  260 I = 0
      ERROR = 0.
      DO 270 IA=1,MX
      NBB = NL(IA)-NU(IA)+1
      DO 270 IB=1,NBB
      I = I+1
      I1 = I-1
      I2 = I+1
      IF(((IA.LT.MXBI).OR.(IA.GT.MXBO)).AND.(IB.EQ.1)) I1 = I1+NBB
      IF(((IA.LT.MXBI).OR.(IA.GT.MXBO)).AND.(IB.EQ.NBB)) I2 = I2-NBB
      I3 = I-NL(IA-1)+NU(IA)-1
      I4 = I+NL(IA)-NU(IA+1)+1
      CHANGE = W*(K(I)-U(I)+A(I,1)*U(I1)+A(I,2)*U(I2)+A(I,3)*U(I3)+
     1   A(I,4)*U(I4))
      ERROR = AMAX1(ERROR,ABS(CHANGE))
      U(I) = U(I)+CHANGE
      IF(NULAKI.EQ.0) GOTO 270
      IF(IA.EQ.1) I3=0
      IF(IA.EQ.MX) I4=0
      WRITE (6,1460) IA,I,(A(I,J),J=1,4),I1,I2,I3,I4,K(I)
  270 CONTINUE
      NULAKI = 0
```

```
      IF(ERPRT.NE.0) WRITE (6,1510) ERROR
      IF(ERROR.GT.TOLER) GO TO 260
      IF(STRFN.NE.0) WRITE(6,1520)
      LAST = 0
      DO 280 IA=1,MX
      FIRST = LAST+1
      LAST = FIRST+NL(IA)-NU(IA)
      IU(IA)=FIRST
      IL(IA)=LAST
  280 IF(STRFN.NE.0) WRITE (6,1530) (U(I),I=FIRST,LAST)
      RETURN
 1450 FORMAT (88H1  IA     I      A(I,1)     A(I,2)     A(I,3)     A(I,4)
     111      I2      I3      I4       K(I)   )
 1460 FORMAT (2X,I3,I6,4F10.5,4I7,F10.5)
 1500 FORMAT (7H WMAX =,F9.6,5X,6HWMIN =,F9.6,5X,6HLMAX =,F9.6,5X,
     1   6HLMIN =,F9.6)
 1510 FORMAT ( 8H ERROR =,F11.8)
 1520 FORMAT (1H1,10X,22HSTREAM FUNCTION VALUES)
 1530 FORMAT (2X,10F13.8)
      END
```

```
$IBFTC SLAXVL

      SUBROUTINE SLAXVL
      REAL K,K1,K2,K3,K4,K5,LMAX,LMIN
      INTEGER SRW,FIRST,CASE,BLDATA,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,SURVEL
      COMMON SRW,PITCH,CHORD,STGR,THETAI,THETAO,DTLR,RI,ALUI,ALLI,
     1   RO,ALUO,ALLO,MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT,
     2   BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,SURVEL,
     3   ZU(50),XSPU(50),ZL(50),XSPL(50),W,WR,TOLER,BDA,BDD,
     4   U(2500),A(2500,4),K(2500),
     5   DXDZU(100),DXDZL(100),SLUPE(50),EMU(50),SLLPE(50),EML(50),
     6   XU(100),XL(100),NU(100),NL(100),UINT(11),XINT(11),
     7   HA,HB,NXN,MXBIM1,MXBOP1,JU,JL,HU,HL,ID,NBBO,NBUO,NCH,
     8   IBTE1,IBTE2,ITE2,HALTE,HANTE
      DIMENSION WZ(2500),WX(2500),V(2500),THETA(2500),SL(1100)
      DIMENSION WZU(100),WZL(100),WXU(100),WXL(100),ZXU(100),ZXL(100),
     1   THETAU(100),THETAL(100),WU(100),WL(100),
     2   USP(100),ZPL(100),IU(100),IL(100),AAA(100)
      EQUIVALENCE (A(1,1),WZ(1)),(A(1,2),WX(1)),(A(1,3),V(1)),
     1   (A(1,4),THETA(1)),(K(1501),SL(1))
      EQUIVALENCE (K(1),WZU(1)),(K(101),WZL(1)),(K(201),WXU(1)),
     1   (K(301),WXL(1)),(K(401),ZXU(1)),(K(501),ZXL(1)),
     2   (K(601),THETAU(1)),(K(701),THETAL(1)),(K(801),WU(1)),
     3   (K(901),WL(1)),(K(1001),USP(1)),(K(1101),ZPL(1)),
     4   (K(1201),IU(1)),(K(1301),IL(1)),(K(1401),AAA(1))
      DIMENSION XBB(52),WZSP(52),KKK(24),P(11)
      DATA (KKK(J),J=4,24,2)/11*1H*/
C
C    CALCULATE STREAMLINE LOCATION -- UPSTREAM
C
      I1=1
      I = 0
      DELINT = 1./FLOAT(NINT)
      NBB = NL(1)+1
      XBB(1) = 0.
      ZPL(1) = HA*FLOAT(1-MXBI)
      DO 318 IA=2,MX
  318 ZPL(IA) = ZPL(IA-1)+HA
      IF(SLCRD.NE.0) WRITE(6,1550)
      DO 320 IB =1,NBB
  320 XBB(IB+1) = XBB(IB)+HB
      DO 350 IA = 1,MXBIM1
      UINT(1) = AINT(U(I+1)/DELINT)*DELINT
      IF(U(I+1).GT.0.) UINT(1) = UINT(1)+DELINT
      DO 330 JB=2,NINT
  330 UINT(JB) = UINT(JB-1)+DELINT
      DO 340 IB=1,NBB
      I = I+1
  340 USP(IB) = U(I)
      NSP = NBB+1
      USP(NSP) = USP(1)+1.
      CALL SPLINT(USP,XBB,NSP,UINT,NINT,XINT)
```

```
      CALL SPLINE(XBB,USP,NSP,WZ(I1),AAA)
      I1=I1+NL(IA)+1
      IF(SLCRD.NE.0) WRITE(6,1540) ZPL(IA),(UINT(J),XINT(J),J=1,NINT)
      DO 345 JB=1,NINT
      J = MX*(JB-1)+IA
      SL(J) = XINT(JB)
  345 CONTINUE
      J1 = MX*NINT+IA
      SL(J1) = SL(J)
  350 CONTINUE
C
C   CALCULATE STREAMLINE LOCATION -- BLADE
C
      JU=MXBI-1
      NINT = NINT+1
      UINT(1) = 0.
      DO 360 JB=2,NINT
  360 UINT(JB) = UINT(JB-1)+DELINT
      USP(1) = 0.
      DO 380 IA=MXBI,MXBO
      XBB(1) = XU(IA)
      NBB = NL(IA)-NU(IA)+1
      XBB(2) = FLOAT(NU(IA))*HB
      DO 370 IB = 1,NBB
      I = I+1
      USP(IB+1) = U(I)
  370 XBB(IB+1) = FLOAT(IB-1)*HB+XBB(2)
      NSP = NBB+2
      USP(NSP) = 1.
      XBB(NSP) = XL(IA)
      CALL SPLINT(USP,XBB,NSP,UINT,NINT,XINT)
      CALL SPLINE(XBB,USP,NSP,WZSP,AAA)
      DO 375 IB=1,NBB
      WZ(I1)=WZSP(IB+1)
  375 I1=I1+1
      JU=JU+1
      WZU(JU)=WZSP(1)
      WZL(JU)=WZSP(NSP)
      IF(SLCRD.NE.0) WRITE(6,1540) ZPL(IA),(UINT(J),XINT(J),J=1,NINT)
      DO 380 JB=1,NINT
      J = MX*(JB-1)+IA
      SL(J) = XINT(JB)
  380 CONTINUE
C
C   CALCULATE STREAMLINE LOCATION -- DOWNSTREAM
C
      NINT = NINT-1
      NBB = NL(MXBOP1)-NU(MXBOP1)+1
      NSP = NBB+1
      XBB(1) = STGR
      XBB(2) = STGR+HU
      DO 390 IB=3,NBB
  390 XBB(IB) = XBB(IB-1)+HB
      XBB(NSP) = STGR+PITCH
      DO 420 IA=MXBOP1,MX
      UINT(1) = AINT(U(I+1)/DELINT)*DELINT
```

```
      IF(U(I+1).GT.0.) UINT(1) = UINT(1)+DELINT
      DO 400 JB=2,NINT
  400 UINT(JB) = UINT(JB-1)+DELINT
      DO 410 IB=1,NBB
      I = I+1
  410 USP(IB) = U(I)
      USP(NSP) = USP(1)+1.
      CALL SPLINT(USP,XBB,NSP,UINT,NINT,XINT)
      CALL SPLINE(XBB,USP,NSP,WZ(I1),AAA)
      I1=I1+NL(IA)-NU(IA)+1
      IF(SLCRD.NE.0) WRITE(6,1540) ZPL(IA),(UINT(J),XINT(J),J=1,NINT)
      DO 415 JB=1,NINT
      J = MX*(JB-1)+IA
      SL(J) = XINT(JB)
  415 CONTINUE
      J1 = MX*NINT+IA
      SL(J1) = SL(J)
  420 CONTINUE
C
C   PLOT STREAMLINES
C
      IF(SLPLT.EQ.0) GO TO 480
C   CALCULATE PLOTTING PARAMETERS
      ZMIN = ZPL(1)
      XMAX = XL(1)
      XMIN = XU(1)
      DO 430 IA = 2,MX
      XMAX = AMAX1(XMAX,XL(IA))
  430 XMIN = AMIN1(XMIN,XU(IA))
      DX = XMAX-XMIN
      XFACT = 2.
  440 IF(DX.GT.10.) GO TO 450
      DX = DX*10.
      XFACT = XFACT+1.
      GO TO 440
  450 IF(DX.LE.100.) GO TO 460
      DX = DX/10.
      XFACT = XFACT-1.
      GO TO 450
  460 DX = AINT(DX+1.)
      DZ = AINT(5.*DX/3.)
      ZFACT = XFACT
      ZMIN = AINT(ZMIN*10.**ZFACT)
      XMIN = AINT(XMIN*10.**XFACT)
      KKK(1) = 49
      KKK(2) = NINT+1
      P(1) = 1.
      P(5) = 0.
      P(6) = 6.-ZFACT
      P(7) = ZMIN
      P(8) = DZ
      P(9) = 6.-XFACT
      P(10) = XMIN
      P(11) = DX
      KKK(3) = MX
      WRITE(6,1530)
```

```
      CALL PLOTMY (ZPL,SL,KKK,P)
      WRITE (6,1560)
      ZPL(1) = HA*FLOAT(1-MXBI)
      DO 470 IA=2,MX
  470 ZPL(IA)=ZPL(IA-1)+HA
  480 IF(ARPRT.EQ.0) RETURN
      WRITE (6,1600)
      LAST=0
      DO 490 IA=1,MX
      FIRST=LAST+1
      LAST = FIRST+NL(IA)-NU(IA)
  490 WRITE (6,1020) (WZ(I),I=FIRST,LAST)
      WRITE (6,1610)
      WRITE (6,1020) (WZU(IA), IA=MXBI,MXBO)
      WRITE (6,1620)
      WRITE (6,1020) (WZL(IA), IA=MXBI,MXBO)
      RETURN
 1020 FORMAT (1X,8G16.7)
 1530 FORMAT (2HPT,50X,16HSTREAMLINE PLOTS  )
 1540 FORMAT (1X,7G18.7/(19X,6G18.7))
 1550 FORMAT (1H1,26X,22HSTREAMLINE COORDINATES///7X,8HZ COORD.,
     1    3(9X,10HSTREAM FN.,9X,8HX COORD.)//)
 1560 FORMAT (2HPL,35X,86HSTREAMLINES ARE PLOTTED WITH X-DIMENSION ACROS
     1S THE PAGE AND Z-DIMENSION DOWN THE PAGE)
 1600 FORMAT(1HO, 8HWZ ARRAY)
 1610 FORMAT(1HO, 9HWZU ARRAY)
 1620 FORMAT(1HO, 9HWZL ARRAY)
      END
```

```
$IBFTC TASVEL

      SUBROUTINE TASVEL
      REAL K,K1,K2,K3,K4,K5,LMAX,LMIN
      INTEGER SRW,FIRST,CASE,BLDATA,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,SURVEL
      COMMON SRW,PITCH,CHORD,STGR,THETAI,THETAO,DTLR,RI,ALUI,ALLI,
     1   RO,ALUO,ALLO,MXBI,MXBO,MX,NBBI,NUSP,NLSP,NINT,
     2   BLDATA,NULAKI,ERPRT,STRFN,SLCRD,SLPLT,ARPRT,INTVEL,SURVEL,
     3   ZU(50),XSPU(50),ZL(50),XSPL(50),W,WR,TOLER,BDA,BDD,
     4   U(2500),A(2500,4),K(2500),
     5   DXDZU(100),DXDZL(100),SLUPE(50),EMU(50),SLLPE(50),EML(50),
     6   XU(100),XL(100),NU(100),NL(100),UINT(11),XINT(11),
     7   HA,HB,NXN,MXBIM1,MXBOP1,JU,JL,HU,HL,ID,NBBO,NBUO,NCH,
     8   IBTE1,IBTE2,ITE2,HA1TE,HANTE
      DIMENSION WZ(2500),WX(2500),V(2500),THETA(2500),SL(1100)
      DIMENSION WZU(100),WZL(100),WXU(100),WXL(100),ZXU(100),ZXL(100),
     1   THETAU(100),THETAL(100),WU(100),WL(100),
     2   USP(100),ZPL(100),IU(100),IL(100),AAA(100)
      EQUIVALENCE (A(1,1),WZ(1)),(A(1,2),WX(1)),(A(1,3),V(1)),
     1   (A(1,4),THETA(1)),(K(1501),SL(1))
      EQUIVALENCE (K(1),WZU(1)),(K(101),WZL(1)),(K(201),WXU(1)),
     1   (K(301),WXL(1)),(K(401),ZXU(1)),(K(501),ZXL(1)),
     2   (K(601),THETAU(1)),(K(701),THETAL(1)),(K(801),WU(1)),
     3   (K(901),WL(1)),(K(1001),USP(1)),(K(1101),ZPL(1)),
     4   (K(1201),IU(1)),(K(1301),IL(1)),(K(1401),AAA(1))
C
C  CALCULATE TANGENTIAL VELOCITY COMPONENTS
C
C  START AT IA = 1 (CASES 1,2,3)
C
      CASE =1
      JU = 0
      JL = 0
      KK1 = 0
      KN = 0
      IB = 0
      IA1 = 1
      HA1 = HA
      IAN = MXBI
      I = NBBI+1
C  END ON UPPER SURFACE (CASE 1)
  510 IF(IB.GT.NL(1)) GO TO 600
      DO 530 IA=IAN,MXBO
  530 IF(NU(IA).GT.IB) GO TO 540
      CASE = 2
      GO TO 550
  540 IAN = IA
      IA = IAN-1
      X1 = FLOAT(IB)*HB
      CALL INTPL(XU,IA,X1,HAN,1,K4,ZU,XSPU,ALUI,ALUO,NUSP,SLUPE,EMU,
     1   RI,RO,CHORD,STGR,PITCH,1.,HA,HB,MXBI,MXBO)
      GO TO 770
```

```
C  END AT IA = MX (CASE 2)
  550 IF(IB.NE.IBTE1) GO TO 554
      IAN = MXBO
      HAN = HANTE
      KN = 1
      GO TO 770
  554 DO 555 IA=MXBI,MXBO
  555 IF(IB.GT.NL(IA)) GO TO 560
      IF(IB.GT.NL(1)) GO TO 600
      IAN = MX
      HAN = HA
      GO TO 770
C  END ON LOWER SURFACE (CASE 3)
  560 CASE = 3
      IF(IB.GT.NL(1)) GO TO 600
      DO 570 IA=MXBI,MXBO
  570 IF(NL(IA).LT.IB) GO TO 580
      IA=MXBO
  580 IAN = IA
      IA = IAN-1
      X1 = FLOAT(IB)*HB
      CALL INTPL (XL,IA,X1,HAN,1,K4,ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML,
     1   RI,RO,CHORD,STGR,PITCH,-1.,HA,HB,MXBI,MXBO)
      KN = 1
      GO TO 770
C  START ON LOWER SURFACE (CASE 4)
  600 CASE = 4
      KK1=1
  620 DO 630 IA=MXBI,MXBO
  630 IF(NL(IA+1).GT.NL(IA)) GO TO 640
      CASE=5
      GO TO 680
  640 IB = NL(IA)+1
      IA1 = IA
  650 DO 660 IA=IA1,MXBO
  660 IF(NL(IA+1).GE.IB) GO TO 670
      CASE = 5
      GO TO 680
  670 IA1 = IA
      IA = IA1+1
      X1 = FLOAT(IB)*HB
      I = IL(IA)+IB-NL(IA)
      CALL INTPL (XL,IA,X1,HA1,0,K3,ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML,
     1   RI,RO,CHORD,STGR,PITCH,-1.,HA,HB,MXBI,MXBO)
      GO TO 740
C  START ON UPPER SURFACE (CASE 5)
  680 DO 690 IA=MXBI,MXBO
  690 IF(NU(IA+1).LT.NU(IA)) GO TO 700
      CASE = 6
      GO TO 765
  700 IB = NU(IA)-1
      IA1 = IA
      KK1=0
  710 DO 720 IA=IA1,MXBO
  720 IF(NU(IA+1).LE.IB) GO TO 730
      CASE = 6
```

```
      GO TO 765
  730 IA1 = IA
      IA = IA +1
      X1 = FLOAT(IB)*HB
      IF(IB.EQ.NU(MX)) X1=STGR
      I = IU(IA)+IB-NU(IA)
      CALL INTPL (XU,IA,X1,HA1,0,K3,ZU,XSPU,ALUI,ALUO,NUSP,SLUPE,EMU,
     1   RI,RO,CHORD,STGR,PITCH,1.,HA,HB,MXBI,MXBO)
C  CASES 4 AND 5
  740 IAN=IA1+1
      DO 750 IA=IAN,MXBO
  750 IF(NL(IA).LT.IB) GO TO 760
      IF(IB.NE.IBTE1) GO TO 755
      IAN = MXBO
      HAN = HANTE
      KN = 1
      GO TO 770
C  END AT IA = MX
  755 IAN = MX
      HAN = HA
      KN = 0
      GO TO 770
C  END ON LOWER SURFACE
  760 IAN = IA
      IA = IA-1
      KN = 1
      CALL INTPL(XL,IA,X1,HAN,1,K4,ZL,XSPL,ALLI,ALLO,NLSP,SLLPE,EML,
     1   RI,RO,CHORD,STGR,PITCH,-1.,HA,HB,MXBI,MXBO)
      GO TO 770
  765 IF(IBTE2.EQ.1000) GO TO 780
      IA1 = MXBO-1
      HA1 = HA1TE
      I = ITE2
      IAN = MX
      HAN = HA
      IB = IBTE2
      KK1 =1
  770 CALL VELOC (U,ZPL,IA1,IAN,HA1,HAN,I,IB,MX,NXN,NBBO,JU,JL,KK1,KN,
     1   USP,WX,WXU,ZXU,WXL,ZXL,NU,NL,AAA)
      IB = IB+1
      I = I+1
      IF(CASE.EQ.5) IB=IB-2
      GO TO (510,550,560,650,710,780),CASE
  780 DO 790 I=1,NXN
  790 WX(I)=-WX(I)
      DO 800 I=1,JU
  800 WXU(I)=-WXU(I)
      DO 810 I=1,JL
  810 WXL(I)=-WXL(I)
      CALL SORTXY(ZXL,WXL,JL)
C
C   END OF TANGENTIAL VELOCITY CALCULATION
C
      IF(ARPRT.EQ.0) GO TO 830
      WRITE (6,1630)
      LAST=0
```

```
      DO 820 IA=1,MX
      FIRST=LAST+1
      LAST=FIRST+NL(IA)-NU(IA)
  820 WRITE(6,1020)  (WX(I), I=FIRST,LAST)
      WRITE (6,1640)
      WRITE(6,1020)  (ZXU(I),WXU(I) , I=1,JU)
      WRITE (6,1650)
      WRITE(6,1020)   (ZXL(I),WXL(I) ,I=1,JL)
C
C   CALCULATE VELOCITIES AND ANGLES AT INTERIOR POINTS
C
  830 IF(INTVEL.EQ.0) GO TO 870
      DO 850 I=1,NXN
      V(I) = SQRT(WX(I)**2+WZ(I)**2)
      IF(WZ(I).EQ.0.) GO TO 840
      THETA(I) = ATAN(WX(I)/WZ(I))*57.29577
      GO TO 850
  840 THETA(I)=90.0
  850 CONTINUE
      WRITE (6,1660)
      LAST=0
      DO 860 IA=1,MX
      FIRST=LAST+1
      LAST=FIRST+NL(IA)-NU(IA)
  860 WRITE(6,1670)IA, (V(I),THETA(I) ,I=FIRST,LAST)
C
C SURFACE VELOCITIES BASED ON AXIAL COMPONENTS
C
  870 IF(SURVEL.EQ.0) RETURN
      WRITE (6,1680)
      THETAU(MXBI)=90.0
      THETAL(MXBI)=-90.0
      WU(MXBI) = 0.
      WL(MXBI) = 0.
      THETAU(MXBO)=-90.0
      THETAL(MXBO)=+90.0
      WU(MXBO) = 0.
      WL(MXBO) = 0.
      MXBIP1=MXBI+1
      MXBOM1=MXBO-1
      DO 880 IA=MXBIP1,MXBOM1
      TANTHU=DXDZU(IA)
      THETAU(IA)=ATAN(TANTHU)*57.29577
      WU(IA)=WZU(IA)*SQRT(1.+TANTHU*TANTHU)
      TANTHL=DXDZL(IA)
      THETAL(IA)=ATAN(TANTHL)*59.29577
  880 WL(IA)=WZL(IA)*SQRT(1.+TANTHL*TANTHL)
      WRITE (6,1690)  (ZPL(IA),WU(IA),THETAU(IA),WZU(IA),WL(IA),
     1   THETAL(IA),WZL(IA) , IA=MXBI,MXBO)
C
C   CALCULATE SURFACE VELOCITIES BASED ON TANGENTIAL COMPONENTS
C
      CALL BLDDER(ZU,XSPU,SLUPE,EMU,NUSP,RI,ALUI,RO,ALUO,CHORD,STGR,
     1 PITCH,1.,JU,ZXU,MXBI,HA,DXDZU)
      CALL BLDDER(ZL,XSPL,SLLPE,EML,NLSP,RI,ALLI,RO,ALLO,CHORD,STGR,
     1 PITCH,-1.,JL,ZXL,MXBI,HA,DXDZL)
```

```
C   UPPER SURFACE
      WRITE (6,1700)
      THETAU(1)=90.0
      WU(1)=ABS(WXU(1))
      THETAU(JU) =-90.0
      WU(JU)=ABS(WXU(JU))
      JUM1=JU-1
      DO 900 I=2,JUM1
      TANTHU=DXDZU(I)
      THETAU(I)=ATAN(TANTHU)*57.29577
      IF(TANTHU.EQ.0.) GO TO 890
      WU(I)= ABS(WXU(I))*SQRT(1.+1./(TANTHU*TANTHU))
      GO TO 900
  890 WU(I) = 0.
  900 CONTINUE
      WRITE (6,1710)  (ZXU(I),WU(I),THETAU(I),WXU(I) , I=1,JU)
C   LOWER SURFACE
      WRITE (6,1720)
      DO 920 I=1,JL
      TANTHL= DXDZL(I)
      THETAL(I)=ATAN(TANTHL)*57.29577
      IF (TANTHL.EQ.0.) GO TO 910
      WL(I)=ABS(WXL(I))*SQRT(1.+1./(TANTHL*TANTHL))
      GO TO 920
  910 WL(I) = 0.
  920 CONTINUE
      WRITE (6,1710)  (ZXL(I),WL(I),THETAL(I),WXL(I) , I=1,JL)
      RETURN
 1020 FORMAT (1X,8G16.7)
 1630 FORMAT(1H1, 8HWX ARRAY)
 1640 FORMAT(1H ,4(6X,3HZXU,13X,3HWXU,7X))
 1650 FORMAT(1H ,4(6X,3HZXL,13X,3HWXL,7X))
 1660 FORMAT (1H1,40X,34HVELOCITIES AT INTERIOR MESH POINTS )
 1670 FORMAT (1HL,3HIA=,I2,5(24H   VELOCITY   ANGLE(DEG))/(3X,
     1   5(G15.4,F9.2)))
 1680 FORMAT (1H1,35X,44HSURFACE VELOCITIES BASED ON AXIAL COMPONENTS /
     1   25X,16HUPPER    SURFACE,26X,16HLOWER    SURFACE/7X,1HZ,2(10X,
     2   8HVELOCITY,3X,10HANGLE(DEG),5X,2HWZ,4X))
 1690 FORMAT (1H ,2G13.4,F9.2,G15.4,G18.4,F9.2,G15.4)
 1700 FORMAT (51H1 SURFACE VELOCITIES BASED ON TANGENTIAL COMPONENTS/
     1   25X,13HUPPER SURFACE/7X,1HZ,10X,8HVELOCITY,3X,10HANGLE(DEG),
     2   5X,2HWX)
 1710 FORMAT (1H ,2G13.4,F9.2,G15.4)
 1720 FORMAT (25X,13HLOWER SURFACE/7X,1HZ,10X,8HVELOCITY,3X,
     1   10HANGLE(DEG),5X,2HWX)
      END
```

Subroutine BLDCRD

BLDCRD obtains the x-coordinates and the slopes of the blade surfaces corresponding to the given z-coordinates. BLDCRD may be used in two ways, either to obtain the information at all vertical mesh lines in one call, as when called by COEF directly, or at a single specified point, as when called by INTPL. The value of NCH is the number of points at which output is desired. Since BLDCRD is used for either the upper or lower blade surface, SURF is used as a code to determine which surface is desired. SURF = 1. for the upper surface and SURF = -1. for the lower surface.

The entire blade surface is defined by the leading- and trailing-edge radii and by two cubic spline curves (upper and lower surfaces), which are piecewise cubic polynomials. The procedure then is to scan the spline points to determine which interval the z-coordinate (ZINT) is in, and then to calculate the x-coordinate and derivative, both of which are specified analytically.

The input variables are as follows:

Z	array of z-coordinates of spline points for blade surface (upper or lower)
XSP	array of x-coordinate of spline points for blade surface (upper or lower)
SLOPE	array of slopes at spline points for blade surface (upper or lower)
EM	array of second derivatives at spline points for blade surface (upper or lower)
NSP	number of spline points on blade surface (upper or lower)
RI	see figure 8 (p. 14)
ALI	either ALUI or ALLI (see fig. 8)
RO	see figure 8
ALO	either ALUO or ALLO (see fig. 8)
CHORD	see figure 8
STGR	see figure 8
PITCH	see figure 8
SURF	code to indicate upper or lower surface, SURF = 1., for upper surface and, SURF = -1., for lower surface
NCH	number of points for which output is desired
ZINT	used as input only when NCH = 1, then it is z-coordinate for which corresponding x-coordinate for blade surface is desired
MXBI	same as main program, number of mesh points on line AB

The output variables are as follows:

X — array of x-coordinates at the vertical mesh lines for blade surface, or if NCH = 1, at z = ZINT

DXDZ — array of slopes at same points as X

The internal variables are as follows:

HA — basic mesh spacing in axial (z) direction

IA — index of vertical mesh line

IFST — index of first point considered

ILST — index of last point considered

K — index of spline point

RMZ — difference between z-coordinate of point considered and z-coordinate of center of leading- or trailing-edge radii

SRW — integer variable in common used to obtain output useful in debugging; when SRW = 19, BLDCRD will write out calculated blade coordinates and corresponding slopes

SW — coefficient with value of zero on upper blade surface and 1 on lower blade surface; used to add pitch to computed blade coordinate for lower surface only

ZINT — z-coordinate at which x-coordinate and slope of blade surface are required

```
$IBFTC BLDCRD

      SUBROUTINE BLDCRD(Z,XSP,SLOPE,EM,NSP,RI,ALI,RO,ALO,CHORD,STGR,
     1   PITCH,SURF,X,NCH,ZINT,MXBI,DXDZ)
C
C    SURF = 1.  --  UPPER SURFACE
C    SURF = -1. --  LOWER SURFACE
C
      DIMENSION XSP(NSP),Z(NSP),X(NCH),SLOPE(NSP),EM(NSP),DXDZ(NCH)
      COMMON SRW
      INTEGER SRW
      SW = 0.
      IF(SURF.LT.0.) SW = 1.
      IFST = 1
      ILST = 1
      IF(NCH.EQ.1) GO TO 10
      IFST = MXBI
      ILST = NCH+MXBI-1
      HA = CHORD/FLOAT(NCH-1)
      ZINT = 0.
   10 K = 2
      DO 100 IA=IFST,ILST
   20 IF(ZINT.GT.Z(1)) GO TO 30
      X(IA) = SQRT(ZINT*(2.*RI-ZINT))*SURF+PITCH*SW
      RMZ = RI-ZINT
      IF(IA.NE.IFST) DXDZ(IA) = RMZ/SQRT(RI**2-RMZ**2)*SURF
      ZINT = ZINT+HA
      GO TO 100
   30 IF(ZINT.LE.Z(K)) GO TO 50
      IF(K.GE.NSP) GO TO 60
      K = K+1
      GO TO 30
   50 S = Z(K)-Z(K-1)
      X(IA) = EM(K-1)*(Z(K)-ZINT)**3/6./S+EM(K)*(ZINT-Z(K-1))**3/6./S
     1   +(XSP(K)/S-EM(K)*S/6.)*(ZINT-Z(K-1))+(XSP(K-1)/S-EM(K-1)*S/6.)
     2   *(Z(K)-ZINT)
      DXDZ(IA) = -EM(K-1)*(Z(K)-ZINT)**2/2./S+EM(K)*(Z(K-1)-ZINT)**2/2.
     1   /S+(XSP(K)-XSP(K-1))/S-(EM(K)-EM(K-1))*S/6.
      ZINT = ZINT+HA
      GO TO 100
   60 IF  ((IA.EQ.NCH).AND.(IA.GT.1)) GO TO 70
      X(IA) = STGR+SURF*SQRT((CHORD-ZINT)*(2.*RO-CHORD+ZINT))+PITCH*SW
      RMZ = CHORD-ZINT-RO
      IF(IA.NE.ILST) DXDZ(IA) = RMZ/SQRT(RO**2-RMZ**2)*SURF
      ZINT = ZINT+HA
      GO TO 100
   70 X(IA) = STGR+PITCH*SW
  100 CONTINUE
      IF(SRW.EQ.19) WRITE(6,1000) (X(IA),DXDZ(IA),IA=IFST,ILST)
      RETURN
 1000 FORMAT (1X,54HINTERPOLATED COORDINATES AND SLOPES COMPUTED BY BLDC
     1RD,/(5X,2E16.8))
      END
```

Subroutine BLDDER

BLDDER obtains the slopes of the blade at given z-coordinates. It is used by TASVEL to obtain the blade slopes at each horizontal mesh line. BLDDER is similar to BLDCRD, except that the z-coordinates are an input array and the x-coordinates are not given as output.

The input variables for BLDDER are the same as those for BCDCRD, except that ZINT is not input. Also included are

ZX — array of z-coordinates from the line BG in figure 2 (p. 5) for which slopes for blade surface are desired; these values must be arranged in increasing order

HA — basic mesh spacing in axial direction

The output of BLDDER is

DXDZ — array of slopes at z-coordinates in array ZX

The internal variables are as follows:

IA — index of point in ZX array

K — index of spline point

RMZ — difference between z-coordinate of point considered and z-coordinate of center of leading- or trailing-edge radii

SRW — integer variable in COMMON used to obtain output useful in debugging; when SRW = 20, BLDDER will write out calculated blade slopes

ZINT — z-coordinate from blade leading edge at which blade slope is desired

ZLE — z-coordinate from line AH in figure 2 (p. 5) of leading edge of blade

```
$IBFTC BLDDER

      SUBROUTINE BLDDER(Z,XSP,SLOPE,EM,NSP,RI,ALI,RO,ALO,CHORD,STGR,
     1   PITCH,SURF,NCH,ZX,MXBI,HA,DXDZ)
C
C    SURF = 1.  --  UPPER SURFACE
C    SURF = -1. --  LOWER SURFACE
C
      DIMENSION XSP(NSP),Z(NSP),SLOPE(NSP),EM(NSP),DXDZ(NCH),ZX(NCH)
      COMMON SRW
      INTEGER SRW
   10 K = 2
      DO 100 IA = 1,NCH
      ZINT = ZX(IA)
   20 IF(ZINT.GT.Z(1)) GO TO 30
      RMZ = RI-ZINT
      IF(IA.NE.1.OR.SURF.LT.0.) DXDZ(IA) = RMZ/SQRT(RI**2-RMZ**2)*SURF
      GO TO 100
   30 IF(ZINT.LE.Z(K)) GO TO 50
      IF(K.GE.NSP) GO TO 60
      K = K+1
      GO TO 30
   50 S = Z(K)-Z(K-1)
      DXDZ(IA) = -EM(K-1)*(Z(K)-ZINT)**2/2./S+EM(K)*(Z(K-1)-ZINT)**2/2.
     1   /S+(XSP(K)-XSP(K-1))/S-(EM(K)-EM(K-1))*S/6.
      GO TO 100
   60 RMZ = CHORD-ZINT-RO
      IF((IA.NE.1.AND.IA.NE.NCH).OR.SURF.LT.0.) DXDZ(IA)=RMZ/SQRT(RO**2
     1-RMZ**2)*SURF
      GO TO 100
  100 CONTINUE
      IF(SRW.EQ.20) WRITE(6,1000) (ZX(IA),DXDZ(IA),IA=1,NCH)
      RETURN
 1000 FORMAT (1X,56HZ COORD. AND INTERPOLATED DERIVATIVES COMPUTED BY BL
     1DDER,/(5X,2E16.8))
      END
```

Subroutine INTPL

To compute the terms of the matrix A of equation (A6), it is necessary to obtain the distance along a horizontal mesh line from a mesh point near the blade to the blade itself. This is the quantity h_3 or h_4 in equation (A1). This value is computed by INTPL. Since the equation of the blade surface is known, this amounts to finding the root of an equation. The root is found by INTPL by an iterative procedure, sometimes called the method of false position (falsi reguli). The variables shown in figure 11 correspond to those used in INTPL. After H has been calculated, the actual value on the spline curve (XI) is computed by BLDCRD and a reduced interval is considered so that the curve still crosses the value X1. Then the procedure is repeated on the smaller interval. A few iterations will determine the value of z for which the spline curve crosses the mesh line, and from this, h_3 or h_4 is determined.

The input variables are as follows:

HA basic mesh spacing in axial direction

HB basic mesh spacing in blade-to-blade direction

IA index of vertical mesh line on which mesh point lies

MXBI number of mesh points on line AB

N integer which is zero when h_3 is to be computed and which is 1 when h_4 is to be computed

X array of blade x-coordinates at each vertical mesh line

X1 x-coordinate of mesh point considered (see fig. 8, p. 14)

The remaining input variables are transmitted to BLDCRD. Their definitions are included in the description of this subroutine.

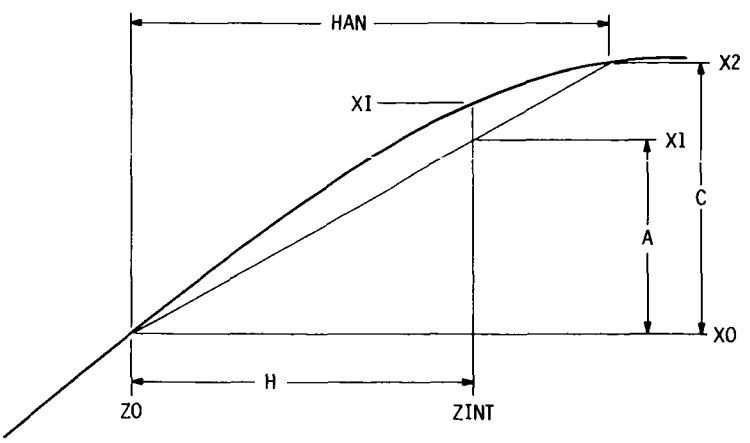


Figure 11. - Notation used in subroutine INTPL $H = \frac{A \times HAN}{C}$.

The output variables are as follows:

H — horizontal distance from mesh point to blade, which is h_3 or h_4 of figure 13

K — real code variable changed to 1 by INTPL

The internal variables are as follows:

A — X1 - XO (see fig. 11)

C — X2 - XO (see fig. 11)

H — distance from ZO to approximate root ZINT determined by linear interpolation (see fig. 11)

HAN — length of interval in which root has been determined to lie (see fig. 11)

IAPN — index of vertical mesh line to left of interval

IAPNM1 — index of vertical mesh line to right of interval

SRW — integer variable in COMMON used to obtain output useful in debugging; when SRW = 19, values of IA, N, HA, X1, X2, XO, and ZO are printed to start, followed by values of H, XI, X1, ZO, and ZBASE for each iteration, and final value of H after convergence

XI — x-coordinate of blade computed by BCDCRD for z = ZINT (see fig. 11)

XO — x-coordinate of blade at right end of interval (see fig. 11)

X2 — x-coordinate of blade at left end of interval (see fig. 11)

ZBASE — z-coordinate of left end of interval for first iteration

ZINT — z-coordinate determined by linear interpolation (see fig. 11)

ZO — z-coordinate of left end of interval (see fig. 11)

```
$IBFTC INTPL

      SUBROUTINE INTPL (X,IA,X1,H,N,K,Z,XSP,ALI,ALO,NSP,SLOPE,EM,RI,RO,
     1   CHORD,STGR,PITCH,SURF,HA,HB,MXBI,MXBO)
      DIMENSION X(100),Z(NSP),XSP(NSP),SLOPE(NSP),EM(NSP)
      COMMON SRW
      INTEGER SRW
      REAL K
      K = 1.
      IAPNM1 = IA+N-1
      IAPN = IA+N
      X0 = X (IAPNM1)
      X2 = X (IAPN)
      HAN = HA
      Z0 = FLOAT(IAPNM1-MXBI)*CHORD/FLOAT(MXBO-MXBI)
      IF(IA.EQ.MXBO) Z0 = CHORD-HA
      H=HA
      IF(SRW.EQ.19) WRITE(6,1010) IA,N,HA,X1,X2,X0,Z0
      IF(Z0.LT.0..OR.Z0.GT.(CHORD-.001*HA)) RETURN
      ZBASE = Z0
      IF(ABS(X1-X0).GT.(.001*HB)) GO TO 10
      A = 0.
      C = H
      GO TO 20
   10 A = X1-X0
      C = X2-X0
   20 H = A/C*HAN
      IF(SRW.EQ.19) WRITE(6,1020) H,XI,X1,Z0,ZBASE
      ZINT = Z0+H
      CALL BLDCRD (Z,XSP,SLOPE,EM,NSP,RI,ALI,RO,ALO,CHORD,STGR,PITCH,
     1   SURF,XI,1,ZINT,MXBI,DX0Z)
      IF(ABS(XI-X1).LE.(HB*.001)) GO TO 40
      IF(A*(XI-X1).LT.0.) GO TO 30
      HAN = H
      X2 = XI
      GOTO10
   30 HAN = HAN-H
      X0 = XI
      Z0 = Z0+H
      GO TO 10
   40 H = Z0+H-ZBASE
      IF(N.EQ.0) H = HA-H
      IF(SRW.EQ.19) WRITE(6,1000) H
      RETURN
 1000 FORMAT (1X,22HH AS COMPUTED BY INTPL /(5X,5E16.8))
 1010 FORMAT (1X,4HIA =,I4,5X,3HN =,I4,5X,4HHA =,E14.6,5X,4HX1 =,E14.6,
     1   4X,4HX2 =,E14.6,4X,4HX0 =,E14.6,4X,4HZ0 =,E14.6)
 1020 FORMAT (1X,3HH =,E14.6,5X,4HXI =,E14.6,5X,4HX1 =,E14.6,5X,4HZ0 =,
     1   E14.6,5X,7HZBASE =,E14.6)
      END
```

Subroutine VELOC

The tangential velocities V_x are computed from $V_x = -\partial u/\partial z$, which require estimating derivatives of the stream function along each horizontal mesh line. Information about the ends of the line, as indicated in figure 12, are calculated by TASVEL, and with this information as input, the required derivatives are computed by VELOC by using SPLINE for the actual calculation of the derivatives based on a cubic spline curve.

The input variables are as follows:

HAN see figure 12

HA1 see figure 12

I mesh point index of second point on line located at vertical mesh line IA2 (see fig. 12)

IAN index of vertical mesh line to right of horizontal mesh line segment (see fig. 12)

IA1 index of vertical mesh line to left of horizontal mesh line segment (see fig. 12)

IB index of horizontal mesh line being considered

KK1 code index to indicate which surface mesh line begins on; lower surface is indicated by KK1 = 1, otherwise KK1 = 0

KN code index to indicate which surface mesh line ends on; lower surface is indicated by KN = 1, otherwise KN = 0

MX total number of vertical mesh lines

NBBO number of mesh lines above mesh line AB for first mesh line below EF (may be negative)

NL array of indexes of first horizontal mesh lines below boundary EFGH, with index of mesh line AB being zero

NU array of indexes of first horizontal mesh line above boundary ABCD, with index of mesh line AB being zero

NXN number of unknown mesh points in region

U array of stream-function values at unknown mesh points

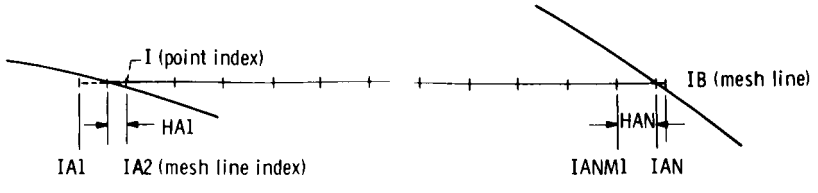


Figure 12. - Notation used in subroutine VELOC.

ZPL array of z-coordinates of vertical mesh lines

The output variables are as follows:

JL increased by 1 for each time one end of mesh line ends on lower blade surface

JU increased by 1 for each time one end of mesh line ends on upper blade surface

WX array of tangential velocity components at unknown mesh points

WXL array of tangential velocity components at each horizontal mesh line along lower blade surface

WXU array of tangential velocity components at each horizontal mesh line along upper blade surface

ZXL array of z-coordinates for which WXL is obtained

ZXU array of z-coordinates for which WXU is obtained

The internal variables are as follows:

AAA array used internally by SPLINE

IA2 index of vertical mesh line at second point from left (see fig. 12)

IANM1 index of vertical mesh line at second point from right (see fig. 12)

I1 mesh point index of point being considered

NSP number of points on horizontal mesh line segment being considered

USP array of stream-function values at points on horizontal mesh line segment

WXSP array of velocities obtained by SPLINE for horizontal mesh line segment

ZA array of z-coordinates of points on horizontal mesh line segment

```
$IBFTC VELOC

      SUBROUTINE VELOC(U,ZPL,IA1,IAN,HA1,HAN,I,IB,MX,NXN,NBBO,JU,JL,
     1   KK1,KN,USP,WX,WXU,ZXU,WXL,ZXL,NU,NL,AAA)
      DIMENSION U(NXN),ZPL(100),WX(NXN),WXU(100),ZXU(100),ZXL(100),
     1   ZA(100),USP(100),WXSP(100),WXL(100),NU(100),NL(100),AAA(100)
C KK1 OR KN = 1, LOWER SURFACE
C KK1 OR KN = 0, UPPER SURFACE
      I1 = I
      IA2 = IA1+1
      IANM1 = IAN-1
      ZA(IA1) = ZPL(IA2)-HA1
      ZA(IAN) = ZPL(IANM1)+HAN
      NSP = IAN-IA1+1
      DO 10 IA=IA2,IANM1
      USP(IA) = U(I1)
      I1=I1+NL(IA)-NU(IA+1)+1
   10 ZA(IA) = ZPL(IA)
      I1 = NXN+IB-NBBO
      USP(IA1)=0.0
      USP(IAN) = 0.
      IF(IA1.EQ.1) USP(1) = U(IB+1)
      IF(KK1.NE.0) USP(IA1) =1.0
      IF(IAN.EQ.MX) USP(IAN) = U(I1)
      IF(KN.NE.0) USP(IAN) = 1.
      CALL SPLINE (ZA(IA1),USP(IA1),NSP,WXSP(IA1),AAA)
      I1 = I
      DO 20 IA=IA2,IANM1
      WX(I1) = WXSP(IA)
   20 I1 = I1+NL(IA)-NU(IA+1)+1
C  TAKE CARE OF FIRST POINT
      IF(IA1.NE.1) GO TO 30
      WX(IB+1) = WXSP(IA1)
      GO TO 50
   30 IF(KK1.NE.0) GO TO 40
      JU = JU+1
      WXU(JU) = WXSP(IA1)
      ZXU(JU) = ZA(IA1)
      GO TO 50
   40 JL = JL+1
      WXL(JL) = WXSP(IA1)
      ZXL(JL) = ZA(IA1)
C  TAKE CARE OF LAST POINT
   50 IF(IAN.NE.MX) GO TO 60
      I1 = NXN+IB-NBBO
      WX(I1) = WXSP(IAN)
      RETURN
   60 IF(KN.NE.0) GO TO 70
      JU = JU+1
      WXU(JU) = WXSP(IAN)
      ZXU(JU) = ZA(IAN)
      RETURN
   70 JL = JL+1
      WXL(JL) = WXSP(IAN)
      ZXL(JL) = ZA(IAN)
      RETURN
      END
```

Subroutine SPLINE

This subroutine is based on the cubic spline curve. SPLINE solves a tridiagonal matrix equation given in reference 6 to obtain the coefficients for the piecewise cubic polynomial function giving the spline fit curve. SPLINE is based on the end condition that the second derivative at either end point is one-half that at the next spline point.

The input variables are as follows:

X array of ordinates

Y array of function values corresponding to X

N number of X and Y values given

The output variables are as follows:

SLOPE array of first derivatives

EM array of second derivatives

If Q = 13 in COMMON, input and output data for SPLINE are printed out. This is useful in debugging.

```
$IBFTC SPLINE

      SUBROUTINE SPLINE (X,Y,N,SLOPE,EM)
      DIMENSION X(N),Y(N),S(100),A(100),B(100),C(100),F(100),W(100),
     1   SB(100),G(100),EM(N),SLOPE(N)
      COMMON Q
      INTEGER Q
      DO 10 I=2,N
   10 S(I)=X(I)-X(I-1)
      NO=N-1
      DO 20 I=2,NO
      A(I)=S(I)/6.
      B(I)=(S(I)+S(I+1))/3.
      C(I)=S(I+1)/6.
   20 F(I)=(Y(I+1)-Y(I))/S(I+1)-(Y(I)-Y(I-1))/S(I)
      A(N)=-.5
      B(1)=1.
      B(N)=1.
      C(1)=-.5
      F(1)=0.
      F(N)=0.
      W(1)=B(1)
      SB(1)=C(1)/W(1)
      G(1)=0.
      DO 30 I=2,N
      W(I)=B(I)-A(I)*SB(I-1)
      SB(I)=C(I)/W(I)
   30 G(I)=(F(I)-A(I)*G(I-1))/W(I)
      EM(N)=G(N)
      DO 40 I=2,N
      K=N+1-I
   40 EM(K)=G(K)-SB(K)*EM(K+1)
      SLOPE(1)=-S(2)/6.*(2.*EM(1)+EM(2))+(Y(2)-Y(1))/S(2)
      DO50 I=2,N
   50 SLOPE(I)=S(I)/6.*(2.*EM(I)+EM(I-1))+(Y(I)-Y(I-1))/S(I)
      IF (Q.EQ.13) WRITE (6,100) N,(X(I),Y(I),SLOPE(I),EM(I),I=1,N)
  100 FORMAT (2X15HNO. OF POINTS =I3/10X5HX     15X5HY     15X5HSLOPE15X5H
     1EM    /(4F20.8))
      RETURN
      END
```

Subroutine SPLN22

This subroutine is the same as SPLINE, except that for the end conditions, the slopes are specified.

The input variables are as follows:

X	array of ordinates
Y	array of function values
YIP	slope at first point
YNP	slope at last point
N	number of X and Y values given

The output variables are as follows:

SCOPE	array of first derivatives
EM	array of second derivatives

If Q = 18 in COMMON, input and output data for SPLN22 are printed out. This is useful in debugging.

```
$IBFTC SPLN22

      SUBROUTINE SPLN22 (X,Y,Y1P,YNP,N,SLOPE,EM)
      DIMENSION X(50),Y(50),S(50),A(50),B(50),C(50),F(50),W(50),SB(50),
     1G(50),EM(50),SLOPE(50)
      COMMON Q
      INTEGER Q
      DO 10 I=2,N
   10 S(I)=X(I)-X(I-1)
      NO=N-1
      DO 20 I=2,NO
      A(I)=S(I)/6.
      B(I)=(S(I)+S(I+1))/3.
      C(I)=S(I+1)/6.
   20 F(I)=(Y(I+1)-Y(I))/S(I+1)-(Y(I)-Y(I-1))/S(I)
      A(N) = S(N)/6.
      B(1)=S(2)/3.
      B(N) = S(N)/3.
      C(1)=S(2)/6.
      F(1)=(Y(2)-Y(1))/S(2)-Y1P
      F(N) = YNP-(Y(N)-Y(N-1))/S(N)
      W(1)=B(1)
      SB(1)=C(1)/W(1)
      G(1)=F(1)/W(1)
      DO 30 I=2,N
      W(I)=B(I)-A(I)*SB(I-1)
      SB(I)=C(I)/W(I)
   30 G(I)=(F(I)-A(I)*G(I-1))/W(I)
      EM(N)=G(N)
      DO 40 I=2,N
      K=N+1-I
   40 EM(K)=G(K)-SB(K)*EM(K+1)
      SLOPE(1)=-S(2)/6.*(2.*EM(1)+EM(2))+(Y(2)-Y(1))/S(2)
      DO50 I=2,N
   50 SLOPE(I)=S(I)/6.*(2.*EM(I)+EM(I-1))+(Y(I)-Y(I-1))/S(I)
      IF (Q.EQ.18) WRITE (6,100) N,(X(I),Y(I),SLOPE(I),EM(I),I=1,N)
      RETURN
  100 FORMAT (2X15HNO. OF POINTS =I3/10X5HZ     15X5HX     10X10HDERIVATIV
     1E10X10H2ND DERIV./(4G20.8))
      END
```

Subroutine SPLINT

This subroutine is based on the cubic spline curve, with the same end conditions as SPLINE. The cubic spline curve is then used for interpolation.

The input variables are as follows:

X array of spline point ordinates

Y array of function values at spline points

N number of X and Y values given

Z array of ordinates at which interpolated function values are desired

MAX number of Z values given

The output variable is as follows:

YINT array of interpolated function values

If Q = 16 in COMMON, or if some element of the z array falls outside of the interval for the x array, then input and output data for SPLINT are printed out. This is useful in debugging.

```
$IBFTC SPLINT

      SUBROUTINE SPLINT (X,Y,N,Z,MAX,YINT)
      DIMENSION X(50),Y(50),S(50),A(50),B(50),C(50),F(50),W(50),SB(50),
     1G(50),EM(50),Z(50),YINT(50)
      COMMON Q
      INTEGER Q
      III = Q
      DO 10 I=2,N
   10 S(I)=X(I)-X(I-1)
      NO=N-1
      DO 20 I=2,NO
      A(I)=S(I)/6.0
      B(I)=(S(I)+S(I+1))/3.0
      C(I)=S(I+1)/6.0
   20 F(I)=(Y(I+1)-Y(I))/S(I+1)-(Y(I)-Y(I-1))/S(I)
      A(N)=-.5
      B(1)=1.0
      B(N)=1.0
      C(1)=-.5
      F(1)=0.0
      F(N)=0.0
      W(1)=B(1)
      SB(1)=C(1)/W(1)
      G(1)=0.0
      DO 30 I=2,N
      W(I)=B(I)-A(I)*SB(I-1)
      SB(I)=C(I)/W(I)
   30 G(I)=(F(I)-A(I)*G(I-1))/W(I)
      EM(N)=G(N)
      DO 40 I=2,N
      K=N+1-I
   40 EM(K)=G(K)-SB(K)*EM(K+1)
      DO 90 I=1,MAX
      K=2
      IF(Z(I)-X(1)) 60,50,70
   50 YINT(I)=Y(1)
      GO TO 90
   60 IF(Z(I).LT.(1.1*X(1)-.1*X(2)))WRITE (6,1000)Z(I)
      Q = 16
      GO TO 85
 1000 FORMAT (17H OUT OF RANGE Z =F10.6)
   65 IF(Z(I).GT.(1.1*X(N)-.1*X(N-1))) WRITE (6,1000)Z(I)
      Q = 16
      K=N
      GO TO 85
   70 IF(Z(I)-X(K)) 85,75,80
   75 YINT(I)=Y(K)
      GO TO 90
   80 K=K+1
      IF(K-N) 70,70,65
   85 YINT(I) = EM(K-1)*(X(K)-Z(I))**3/6./S(K)+EM(K)*(Z(I)-X(K-1))**3/6.
```

```
     1/S(K)+(Y(K)/S(K)-EM(K)*S(K)/6.)*(Z(I)-X(K-1))+(Y(K-1)/S(K)-EM(K-1)
     2*S(K)/6.)*(X(K)-Z(I))
  90 CONTINUE
     MXA = MAXO(N,MAX)
     IF(Q.EQ.16) WRITE(6,1010) N,MAX,(X(I),Y(I),Z(I),YINT(I),I=1,MXA)
     Q = III
1010 FORMAT (2X21HNO. OF POINTS GIVEN =,I3,30H, NO. OF INTERPOLATED POI
     1NTS =,I3,/10X5HX     15X5HY     12X11HX-INTERPOL.9X11HY-INTERPOL./(4
     2E20.8))
 100 RETURN
     END
```

Subroutine SORTXY

This subroutine rearranges the NPTS elements of the V array in increasing order. The elements of the W array are moved to maintain the original pair relation; that is, if the fifth element of the V array is moved to the first position of V, the fifth element of W is moved to the first position of W.

```
$IBFTC SORTXY

      SUBROUTINE SORTXY(X,Y,NPTS)
      DIMENSION X(100),Y(100)
  100 N=NPTS
  102 NN=N-1
  104 DO 140 KT=1,NN
      XMIN=X(KT)
      JAD=KT
      JKL=KT+1
  112 DO 120 JK=JKL,N
  114 IF (XMIN-X(JK)) 120,120,116
  116 XMIN=X(JK)
  118 JAD=JK
  120 CONTINUE
  122 YMIN=Y(JAD)
      X(JAD)= X(KT)
      Y(JAD)= Y(KT)
      X(KT)= XMIN
      Y(KT)=YMIN
  140 CONTINUE
       RETURN
      END
```

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, August 23, 1966,
720-03-01-35-22.

APPENDIX A

FINITE-DIFFERENCE APPROXIMATION

An approximate numerical solution for the stream function u can be obtained by finite-difference methods. These methods involve first establishing a rectangular grid of mesh points in the region as shown in figure 3 (p. 6). Then at each point where the value of the stream function is unknown, a finite-difference approximation to equation (1) can be written. Adjacent to the boundary, the boundary conditions are included. If there are n unknown values, n linear equations are obtained in n unknowns. The method used to solve these equations is point successive overrelaxation (SOR). This method is an iterative technique that gives rapid convergence to the solution and is completely described in reference 9.

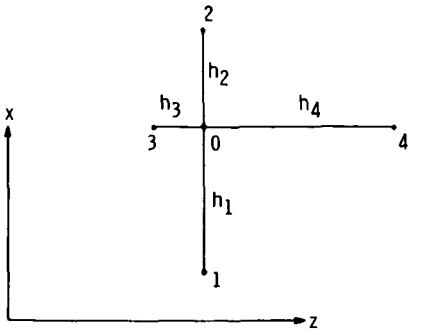


Figure 13. - Notation for adjacent mesh points and mesh spaces.

A typical mesh point with the numbering used to indicate neighboring mesh points is shown in figure 13. The value of the stream function u at 0 is denoted by u_0, and similarly for the neighboring points. It can be shown (ref. 9) that equation (1) can be approximated by

$$u_0 = \sum_{i=1}^{4} a_i u_i \tag{A1}$$

$$a_1 = \frac{h_3 + h_4}{a_0 h_1}$$

$$a_2 = \frac{h_3 + h_4}{a_0 h_2}$$

$$a_3 = \frac{h_1 + h_2}{a_0 h_3}$$

$$a_4 = \frac{h_1 + h_2}{a_0 h_4}$$

$$a_0 = (h_3 + h_4)\left(\frac{1}{h_1} + \frac{1}{h_2}\right) + (h_1 + h_2)\left(\frac{1}{h_3} + \frac{1}{h_4}\right)$$

This equation can be used at all interior mesh points.

Along the boundary where the value of u is unknown, the equation will vary. For example, along the upstream boundary, $\partial u/\partial \eta$ is known, and a finite-difference approximation to $(\partial u/\partial \eta)_{in}$ in equation (3) gives

$$u_0 = u_4 + h_4\left(\frac{\partial u}{\partial \eta}\right)_{in} = u_4 + h_4\left(\frac{\tan \theta_{in}}{s}\right) \tag{A2}$$

Similarly, along the downstream boundary,

$$u_0 = u_3 + h_3\left(\frac{\partial u}{\partial \eta}\right)_{out} = u_3 - h_3\left(\frac{\tan \theta_{out}}{s}\right) \tag{A3}$$

For the points along AB and CD, equations can be derived by using the periodic boundary condition. If the point 0 (fig. 14) is on the boundary between A and B, the point 1 is outside the boundary. However, it is known that $u_1 = u_{1,s} - 1$, where the point 1, s is a distance s above point 1 in the x-direction, as shown in figure 14. Substituting this condition in equation (A1) gives

$$u_0 = a_1 u_{1,s} + \sum_{i=2}^{4} a_i u_i - a_1 \tag{A4}$$

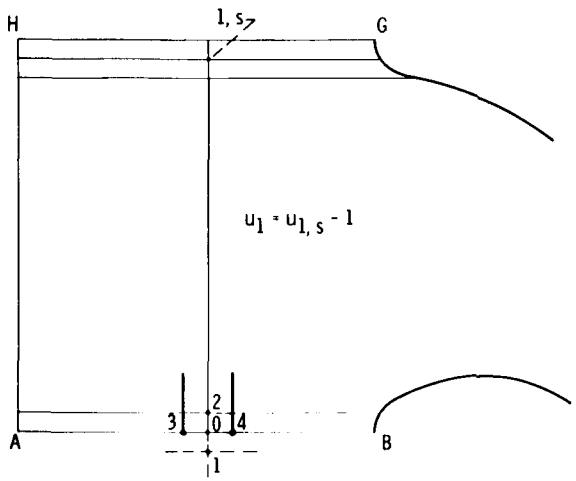


Figure 14. - Mesh point on line AB.

where the a_i are the same as defined in equation (A1). Of course, equation (A4) holds along CD also.

The points along GH need not be considered, since they are just 1 greater than the corresponding point along AB. The equation for the first mesh line below HG must be modified. In this case $u_2 = u_{2,-s} + 1$, where the point 2, -s is a distance s below point 2 in the negative x-direction, as indicated in figure 15. Substituting this condition in equation (A1), gives

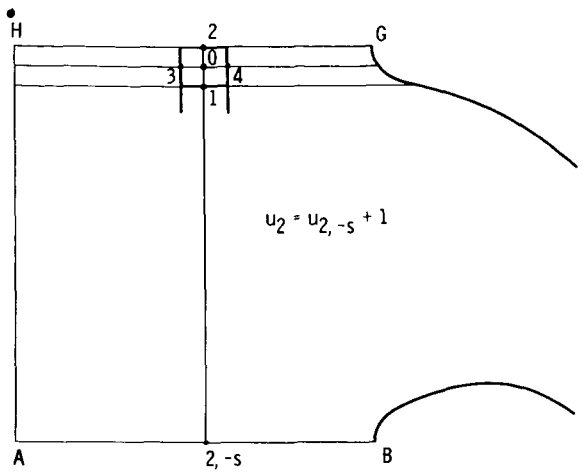


Figure 15. - Mesh point on first line below HG.

$$u_0 = a_1u_1 + a_2u_{2,-s} + a_3u_3 + a_4u_4 + a_2 \qquad \text{(A5)}$$

Equation (A5) also applies to the first mesh line below FE.

One of equations (A1) to (A5) can be applied to each mesh point for which the stream function is unknown in the region of interest, giving the same number of linear equations as there are unknowns. These points where the stream function is unknown will be referred to simply as unknown mesh points.

This system of n equations is represented in matrix form as

$$A\underline{u} = \underline{k} \qquad \text{(A6)}$$

where $\underline{u} = (u_1, \ldots, u_n)^T$ is a vector whose components are the unknown values of the stream function, A is the coefficient matrix of equations (A1) to (A5), and $\underline{k} = (k_1, \ldots, k_n)^T$ is the vector whose components are the known constants of equations (A1) to (A5). Since the coefficient matrix A is irreducibly diagonally dominant, there is a unique solution (ref. 9).

A numerical solution for equation (A6) can be obtained by iterative techniques. These techniques are particularly valuable in solving systems of linear equations of this type, that is, where there are a large number of unknowns, but few terms in each equation. Storage requirements are small, and the roundoff error is minimized. In order to obtain a high rate of convergence, successive overrelaxation (SOR) is used in the program. The equation is given by

$$u_i^{m+1} = u_i^m + \omega\left\{-\sum_{j=1}^{i-1} a_{ij}u_j^{m+1} - \sum_{j=i+1}^{n} a_{ij}u_j^m + k_i - u_i^m\right\} \quad \text{for } i = 1,2, \ldots, n \qquad \text{(A7)}$$

where ω is the overrelaxation factor. The a_{ij} are the elements of the matrix A, and the k_i are the components of the vector $\underline{k}$ of equation (A6). The u_i^o are the initial estimates of the u_i. It can be shown (ref. 9) that equation (A7) will always converge to the solution u of equation (A6) for any initial guess $\underline{u}^o$ if $0 < \omega < 2$. The point is, however, that much more rapid convergence can be obtained for certain values of $\omega > 1$. In fact there exists an optimum value for ω that can be estimated (ref. 9). A method of estimating this optimum value of the overrelaxation factor ω is given in appendix B.

APPENDIX B

THEORETICAL ESTIMATION OF OPTIMUM OVERRELAXATION FACTOR

The optimum value of ω (ref. 9) is given by

$$\omega = \frac{2}{1 + \sqrt{1 - \rho^2(B)}} \tag{B1}$$

where $\rho(B)$ is the spectral radius of the matrix $B = A - I$. In reference 5 it is shown that $\rho^2(B) = \rho(L_1)$, where L_ω is the coefficient matrix for equation (A7), and L_1 is the matrix when $\omega = 1$. Thus, the problem is essentially to estimate $\rho(L_1)$. The simplest method is the power method. Reference 9 shows that $\rho(L_1)$ is a simple eigenvalue, and that the associated eigenvector has positive components. In this case, $\rho(L_1)$ can be estimated by

$$\min_i \left(\frac{u_i^{m+1}}{u_i^m} \right) \leq \rho(L_1) \leq \max_i \left(\frac{u_i^{m+1}}{u_i^m} \right) \tag{B2}$$

and

$$\rho(L_1) = \lim_{m \to \infty} \min_i \left(\frac{u_i^{m+1}}{u_i^m} \right) = \lim_{m \to \infty} \max_i \left(\frac{u_i^{m+1}}{u_i^m} \right) \tag{B3}$$

where u_i^{m+1} is calculated from u_i^m by equation (A7) with $\omega = 1$ and $\underline{k} = \underline{0}$, and $\underline{u}^0$ is an arbitrary vector with positive components. Thus, it would appear that the iteration merely has to be performed until the upper and lower limits are sufficiently close. However, the practical convergence of this method can become extremely slow. What actually happens is that the upper bound converges to a constant value fairly quickly, and that the lower value also converges reasonably to a constant value. However, the values are different. In some cases another 500 iterations were made with no further change in either of the two bounds in the seventh decimal place.

Further experimentation revealed the source of the difficulty. There is rapid convergence in the region between the blades, and much slower convergence in the regions upstream and downstream of the blades, because of the difference in the type of boundary conditions. Convergence is much more rapid where the boundary value is specified

(Dirichlet boundary conditions) than where there is a normal derivative specified (Neumann boundary condition) or where there is a periodic boundary condition. Thus it is almost as if there were two separate regions so that a large change in the upstream flow angle would have a negligible effect on the downstream solution. In estimating $\rho(L_1)$, a spectral radius for the upstream and downstream regions is estimated almost as if the regions were not connected. Eventually the procedure would converge, but it would take an extremely long time for information to be transmitted between the upstream and downstream regions.

In order to test this hypothesis, a special case was run on the computer with only 65 mesh points. The results are shown in figure 16. The upper and lower limits both converged to constant values within only 20 iterations. Both values remained essentially constant for the next 300 iterations! Then, finally, the lower limit started to increase, and after another 300 iterations had come close to the upper limit. The point now is that the upper limit remained unchanged after 20 iterations and is $\rho(L_1)$. Thus, an accurate value of $\rho(L_1)$ was determined after only 20 iterations for this simple case with 65 mesh points. The principle applies to other cases. When the upper limit is observed to become essentially constant, it can be assumed to be $\rho(L_1)$.

For a few cases, experiments with varying values for ω were made. In these few cases it has not been extremely critical to use exactly the theoretically optimum value of ω, although the most rapid convergence was obtained fairly close to the theoretical value.

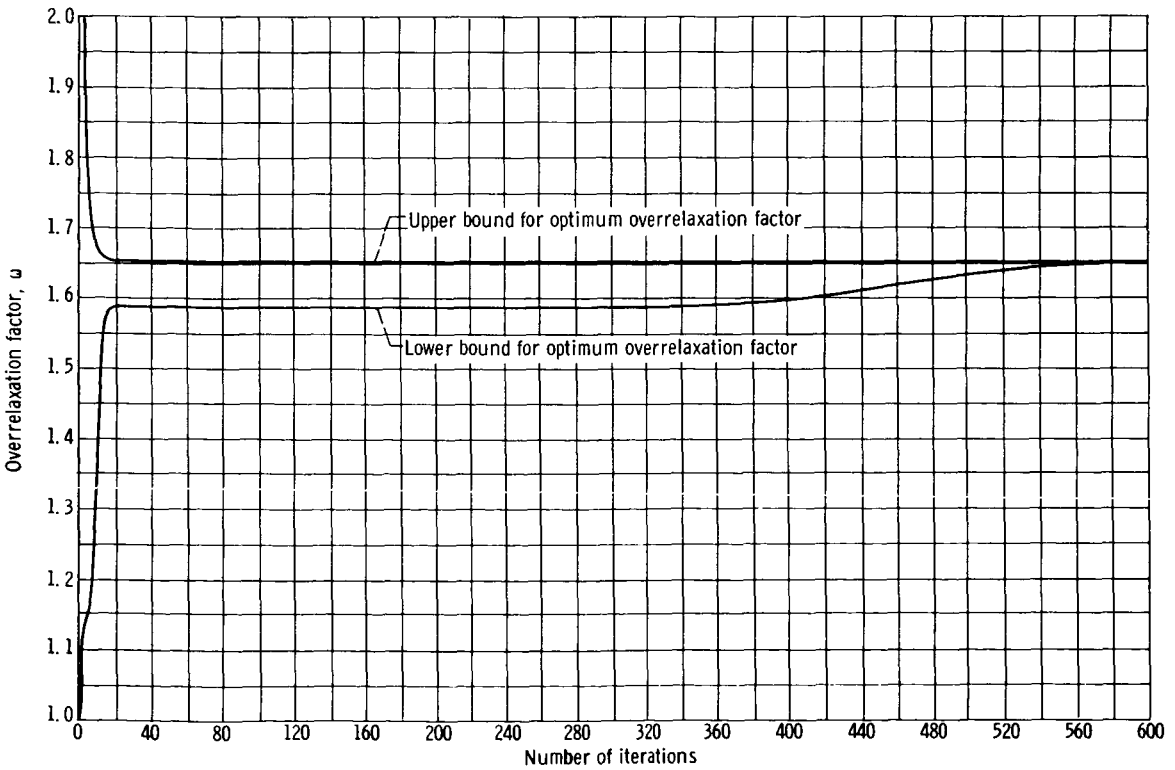


Figure 16. - Bounds for optimum value of overrelaxation factor for special case with 65 mesh points.

Furthermore, even though the upper limit in equation (B3) does converge to the spectral radius $\rho(L_1)$ at a reasonable rate, a considerable amount of computation is still required, and the calculation is comparable to the computation of the solution of equation (A6). Therefore, it is not desirable to compute the spectral radius for every case. Actually, the value of the spectral radius depends primarily on the maximum of the number of unknowns in the upstream region or in the downstream region and secondarily on the shape of the region. Thus, the optimum value of ω can be estimated based on calculations for other similar regions.

REFERENCES

1. Kramer, J. J.: Analysis of Incompressible, Nonviscous Blade-to-Blade Flow in Rotating Blade Rows. ASME Trans., vol. 80, no. 2, Feb. 1958, pp. 263-275.

2. Kramer, James J.; Stockman, Norbert O.; and Bean, Ralph J.: Nonviscous Flow Through a Pump Impeller on a Blade-to-Blade Surface of Revolution. NASA TN D-1108, 1962.

3. Stanitz, John D.; and Ellis, Gaylord O.: Two-Dimensional Compressible Flow in Centrifugal Compressors with Straight Blades. NACA TR 954, 1950.

4. Stanitz, John D.: Two-Dimensional Compressible Flow in Turbomachines with Conic Flow Surfaces. NACA TR 935, 1949.

5. Whitney, Warren J.; Szanca, Edward M.; Moffitt, Thomas P.; and Monroe, Daniel E.: Cold Air Investigation of a Turbine for High-Temperature Engine Application. Pt. I. Turbine Design and Overall Stator Performance. NASA TN D-3751, 1966.

6. Walsh, J. L.; Ahlberg, J. H.; and Nilson, E. N.: Best Approximation Properties of the Spline Fit. J. Math. Mech., vol. 11, 1962, pp. 225-234.

7. Dellner, Lois T.; and Moore, Betty Jo: An Optimized Printer Plotting System Consisting of Complementary 7090 (FORTRAN) and 1401 (SPS) Subroutines. Part II - Systems Programmers Manual. NASA TN D-2175, 1964.

8. Dellner, Lois T.; and Moore, Betty Jo: An Optimized Printer Plotting System Consisting of Complementary 7090 (FORTRAN) and 1401 (SPS) Subroutines. I - Instructions for Users. NASA TN D-2174, 1964.

9. Varga, Richard S.: Matrix Iterative Analysis. Prentice-Hall, Inc., 1962.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Technical information generated in connection with a NASA contract or grant and released under NASA auspices.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

TECHNICAL REPRINTS: Information derived from NASA activities and initially published in the form of journal articles.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities but not necessarily reporting the results of individual NASA-programmed scientific efforts. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546